Join the explorers, builders, and individuals who boldly offer new solutions to old problems. For open source, innovation is only possible because of the people behind it.

Red Hat
Training and Certification

STUDENT WORKBOOK (ROLE)

Red Hat Enterprise Linux 8.0 RH134

RED HAT SYSTEM ADMINISTRATION II

Edition 1

Red Hat

RH134-RHEL8.0-en-1-20190531

Copyright ©2019 Red Hat, Inc.

RED HAT SYSTEM ADMINISTRATION II

Red Hat

RH134-RHEL8.0-en-1-20190531

Copyright ©2019 Red Hat, Inc.

Red Hat Enterprise Linux 8.0 RH134
Red Hat System Administration II
Edition 1 20190531
Publication date 20190531

Authors: Fiona Allen, Adrian Andrade, Herve Quatremain, Victor Costea, Snehangshu Karmakar, Marc Kesler, Saumik Paul

Editor: Philip Sweany, Ralph Rodriguez, David Sacco, Seth Kenlon, Heather Charles

Copyright © 2019 Red Hat, Inc.

The contents of this course and all its modules and related materials, including handouts to audience members, are Copyright © 2019 Red Hat, Inc.

No part of this publication may be stored in a retrieval system, transmitted or reproduced in any way, including, but not limited to, photocopy, photograph, magnetic, electronic or other record, without the prior written permission of Red Hat, Inc.

This instructional program, including all material provided herein, is supplied without any guarantees from Red Hat, Inc. Red Hat, Inc. assumes no liability for damages or legal action arising from the use or misuse of contents or details contained herein.

If you believe Red Hat training materials are being used, copied, or otherwise improperly distributed please e-mail training@redhat.com or phone toll-free (USA) +1 (866) 626-2994 or +1 (919) 754-3700.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, Hibernate, Fedora, the Infinity Logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux® is the registered trademark of Linus Torvalds in the United States and other countries.

Java® is a registered trademark of Oracle and/or its affiliates.

XFS® is a registered trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

The OpenStack® Word Mark and OpenStack Logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Contributors: Achyut Madhusudan, Rob Locke, Rudolf Kastl, Prashant Rastogi, Heider Souza, Michael Phillips, Dallas Spohn

Document Conventions ix

Introduction xi
Red Hat System Administration II ... xi
Orientation to the Classroom Environment ... xii
Controlling Your Systems ... xiii
Internationalization ... xv

1. Improving Command-line Productivity 1
Writing Simple Bash Scripts ... 2
Guided Exercise: Writing Simple Bash Scripts ... 6
Running Commands More Efficiently Using Loops ... 9
Guided Exercise: Running Commands More Efficiently Using Loops ... 15
Matching Text in Command Output with Regular Expressions ... 17
Guided Exercise: Matching Text in Command Output with Regular Expressions ... 26
Lab: Improving Command-line Productivity ... 29
Summary ... 36

2. Scheduling Future Tasks 37
Scheduling a Deferred User Job ... 38
Guided Exercise: Scheduling a Deferred User Job ... 40
Scheduling Recurring User Jobs ... 44
Guided Exercise: Scheduling Recurring User Jobs ... 47
Scheduling Recurring System Jobs ... 50
Guided Exercise: Scheduling Recurring System Jobs ... 53
Managing Temporary Files ... 57
Guided Exercise: Managing Temporary Files ... 60
Quiz: Scheduling Future Tasks ... 64
Summary ... 68

3. Tuning System Performance 69
Adjusting Tuning Profiles ... 70
Guided Exercise: Adjusting Tuning Profiles ... 75
Influencing Process Scheduling ... 77
Guided Exercise: Influencing Process Scheduling ... 81
Lab: Tuning System Performance ... 85
Summary ... 90

4. Controlling Access to Files with ACLs 91
Interpreting File ACLs ... 92
Quiz: Interpreting File ACLs ... 99
Securing Files with ACLs ... 101
Guided Exercise: Securing Files with ACLs ... 105
Lab: Controlling Access to Files with ACLs ... 111
Summary ... 120

5. Managing SELinux Security 121
Changing the SELinux Enforcement Mode ... 122
Guided Exercise: Changing the SELinux Enforcement Mode ... 126
Controlling SELinux File Contexts ... 129
Guided Exercise: Controlling SELinux File Contexts ... 133
Adjusting SELinux Policy with Booleans ... 136
Guided Exercise: Adjusting SELinux Policy with Booleans ... 138
Investigating and Resolving SELinux Issues ... 141
Guided Exercise: Investigating and Resolving SELinux Issues ... 145
Lab: Managing SELinux Security ... 149
Summary ... 155

6. Managing Basic Storage **157**
Adding Partitions, File Systems, and Persistent Mounts 158
Guided Exercise: Adding Partitions, File Systems, and Persistent Mounts 168
Managing Swap Space 172
Guided Exercise: Managing Swap Space 176
Lab: Managing Basic Storage 181
Summary 191

7. Managing Logical Volumes **193**
Creating Logical Volumes 194
Guided Exercise: Creating Logical Volumes 201
Extending Logical Volumes 207
Guided Exercise: Extending Logical Volumes 212
Lab: Managing Logical Volumes 216
Summary 223

8. Implementing Advanced Storage Features **225**
Managing Layered Storage with Stratis 226
Guided Exercise: Managing Layered Storage with Stratis 231
Compressing and Deduplicating Storage with VDO 237
Guided Exercise: Compressing and Deduplicating Storage with VDO 240
Lab: Implementing Advanced Storage Features 245
Summary 255

9. Accessing Network-Attached Storage **257**
Mounting Network-Attached Storage with NFS 258
Guided Exercise: Managing Network-Attached Storage with NFS 263
Automounting Network-Attached Storage 268
Guided Exercise: Automounting Network-Attached Storage 271
Lab: Accessing Network-Attached Storage 277
Summary 284

10. Controlling the Boot Process **285**
Selecting the Boot Target 286
Guided Exercise: Selecting the Boot Target 291
Resetting the Root Password 294
Guided Exercise: Resetting the Root Password 298
Repairing File System Issues at Boot 300
Guided Exercise: Repairing File System Issues at Boot 302
Lab: Controlling the Boot Process 305
Summary 311

11. Managing Network Security **313**
Managing Server Firewalls 314
Guided Exercise: Managing Server Firewalls 322
Controlling SELinux Port Labeling 327
Guided Exercise: Controlling SELinux Port Labeling 330
Lab: Managing Network Security 334
Summary 343

12. Installing Red Hat Enterprise Linux **345**
Installing Red Hat Enterprise Linux 346
Guided Exercise: Installing Red Hat Enterprise Linux 350
Automating Installation with Kickstart 353
Guided Exercise: Automating Installation with Kickstart 362
Installing and Configuring Virtual Machines 365
Quiz: Installing and Configuring Virtual Machines 370
Lab: Installing Red Hat Enterprise Linux 372

Summary 379

13. Comprehensive Review **381**

Comprehensive Review 382
Lab: Fixing Boot Issues and Maintaining Servers 385
Lab: Configuring and Managing File Systems and Storage 393
Lab: Configuring and Managing Server Security 400

DOCUMENT CONVENTIONS

REFERENCES

"References" describe where to find external documentation relevant to a subject.

NOTE

"Notes" are tips, shortcuts or alternative approaches to the task at hand. Ignoring a note should have no negative consequences, but you might miss out on a trick that makes your life easier.

IMPORTANT

"Important" boxes detail things that are easily missed: configuration changes that only apply to the current session, or services that need restarting before an update will apply. Ignoring a box labeled "Important" will not cause data loss, but may cause irritation and frustration.

WARNING

"Warnings" should not be ignored. Ignoring warnings will most likely cause data loss.

INTRODUCTION

RED HAT SYSTEM ADMINISTRATION II

This course is specifically designed for students who have completed Red Hat System Administration I (RH124). Red Hat System Administration II (RH134) focuses on the key tasks needed to become a full time Linux Administrator and to validate those skills via the Red Hat Certified System Administrator exam. This course goes deeper into Enterprise Linux administration including filesystems and partitioning, logical volumes, SELinux, firewalling, and troubleshooting.

COURSE OBJECTIVES

- Expand and extend on skills gained during the Red Hat System Administration I (RH124) course.
- Build skills needed by an RHCSA-certified Red Hat Enterprise Linux system administrator.

AUDIENCE

- This course is singularly designed for students who have completed Red Hat System Administration I (RH124). The organization of topics is such that it is not appropriate for students to use RH134 as a curriculum entry point. Students who have not taken a previous Red Hat course are encouraged to take either System Administration I (RH124) if they are new to Linux or the RHCSA Fast Track course (RH200) if they are experienced with Enterprise Linux administration.

PREREQUISITES

- Having sat the Red Hat System Administration I (RH124) course, or equivalent knowledge.

ORIENTATION TO THE CLASSROOM ENVIRONMENT

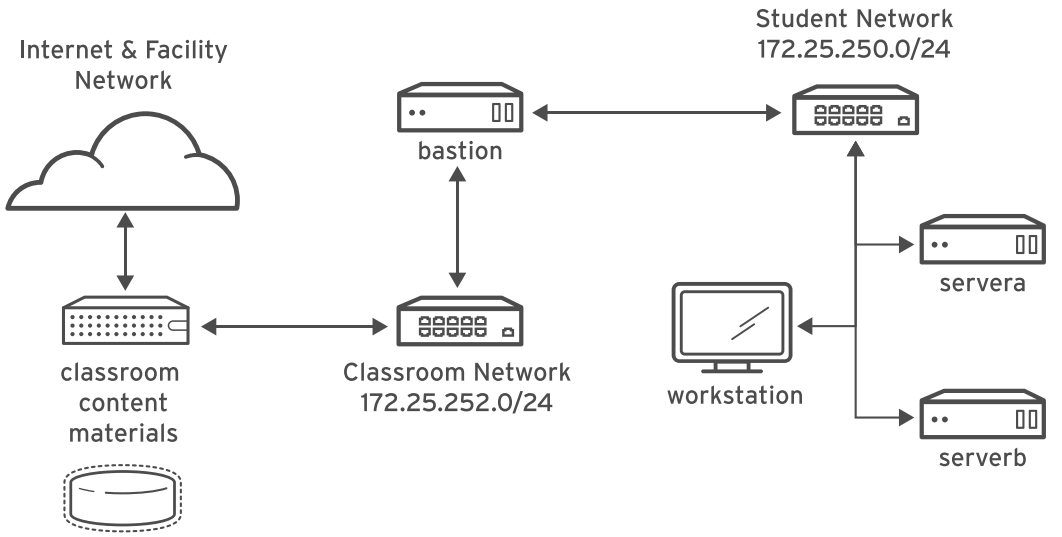

Figure 0.1: Classroom environment

In this course, the main computer system used for hands-on learning activities is `workstation`. Two other machines are also used by students for these activities: `servera`, and `serverb`. All three of these systems are in the `lab.example.com` DNS domain.

All student computer systems have a standard user account, `student`, which has the password `student`. The `root` password on all student systems is `redhat`.

Classroom Machines

MACHINE NAME	IP ADDRESSES	ROLE
bastion.lab.example.com	172.25.250.254	Gateway system to connect student private network to classroom server (must always be running)
workstation.lab.example.com	172.25.250.9	Graphical workstation used for system administration
servera.lab.example.com	172.25.250.10	First server
serverb.lab.example.com	172.25.250.11	Second server

The primary function of `bastion` is that it acts as a router between the network that connects the student machines and the classroom network. If `bastion` is down, other student machines will only be able to access systems on the individual student network.

Several systems in the classroom provide supporting services. Two servers, `content.example.com` and `materials.example.com`, are sources for software and lab materials used in hands-on activities. Information on how to use these servers is provided in the instructions for those activities. These are provided by the `classroom.example.com` virtual machine. Both `classroom` and `bastion` should always be running for proper use of the lab environment.

NOTE

When logging on to `servera` or `serverb` you might see a message concerning the activation of `cockpit`. The message can be ignored.

```
[student@workstation ~]$ ssh student@serverb
Warning: Permanently added 'serverb,172.25.250.11' (ECDSA) to the list
 of known hosts.
Activate the web console with: systemctl enable --now cockpit.socket

[student@serverb ~]$
```

CONTROLLING YOUR SYSTEMS

Students are assigned remote computers in a Red Hat Online Learning classroom. They are accessed through a web application hosted at rol.redhat.com [http://rol.redhat.com]. Students should log in to this site using their Red Hat Customer Portal user credentials.

Controlling the Virtual Machines

The virtual machines in your classroom environment are controlled through a web page. The state of each virtual machine in the classroom is displayed on the page under the ***Online Lab*** tab.

Machine States

VIRTUAL MACHINE STATE	DESCRIPTION
STARTING	The virtual machine is in the process of booting.
STARTED	The virtual machine is running and available (or, when booting, soon will be).
STOPPING	The virtual machine is in the process of shutting down.
STOPPED	The virtual machine is completely shut down. Upon starting, the virtual machine boots into the same state as when it was shut down (the disk will have been preserved).
PUBLISHING	The initial creation of the virtual machine is being performed.
WAITING_TO_START	The virtual machine is waiting for other virtual machines to start.

Depending on the state of a machine, a selection of the following actions is available.

Classroom/Machine Actions

BUTTON OR ACTION	DESCRIPTION
PROVISION LAB	Create the ROL classroom. Creates all of the virtual machines needed for the classroom and starts them. Can take several minutes to complete.
DELETE LAB	Delete the ROL classroom. Destroys all virtual machines in the classroom. **Caution: Any work generated on the disks is lost.**
START LAB	Start all virtual machines in the classroom.
SHUTDOWN LAB	Stop all virtual machines in the classroom.
OPEN CONSOLE	Open a new tab in the browser and connect to the console of the virtual machine. Students can log in directly to the virtual machine and run commands. In most cases, students should log in to the `workstation` virtual machine and use **`ssh`** to connect to the other virtual machines.
ACTION → Start	Start (power on) the virtual machine.
ACTION → Shutdown	Gracefully shut down the virtual machine, preserving the contents of its disk.
ACTION → Power Off	Forcefully shut down the virtual machine, preserving the contents of its disk. This is equivalent to removing the power from a physical machine.
ACTION → Reset	Forcefully shut down the virtual machine and reset the disk to its initial state. **Caution: Any work generated on the disk is lost.**

At the start of an exercise, if instructed to reset a single virtual machine node, click ***ACTION → Reset*** for only the specific virtual machine.

At the start of an exercise, if instructed to reset all virtual machines, click ***ACTION → Reset***

If you want to return the classroom environment to its original state at the start of the course, you can click ***DELETE LAB*** to remove the entire classroom environment. After the lab has been deleted, you can click ***PROVISION LAB*** to provision a new set of classroom systems.

> **WARNING**
> The ***DELETE LAB*** operation cannot be undone. Any work you have completed in the classroom environment up to that point will be lost.

The Autostop Timer

The Red Hat Online Learning enrollment entitles students to a certain amount of computer time. To help conserve allotted computer time, the ROL classroom has an associated countdown timer, which shuts down the classroom environment when the timer expires.

To adjust the timer, click ***MODIFY*** to display the ***New Autostop Time*** dialog box. Set the number of hours until the classroom should automatically stop. Click ***ADJUST TIME*** to apply this change to the timer settings.

INTERNATIONALIZATION

PER-USER LANGUAGE SELECTION

Your users might prefer to use a different language for their desktop environment than the system-wide default. They might also want to use a different keyboard layout or input method for their account.

Language Settings

In the GNOME desktop environment, the user might be prompted to set their preferred language and input method on first login. If not, then the easiest way for an individual user to adjust their preferred language and input method settings is to use the ***Region & Language*** application.

You can start this application in two ways. You can run the command **`gnome-control-center region`** from a terminal window, or on the top bar, from the system menu in the right corner, select the settings button (which has a crossed screwdriver and wrench for an icon) from the bottom left of the menu.

In the window that opens, select ***Region & Language***. Click the ***Language*** box and select the preferred language from the list that appears. This also updates the ***Formats*** setting to the default for that language. The next time you log in, these changes will take full effect.

These settings affect the GNOME desktop environment and any applications such as **`gnome-terminal`** that are started inside it. However, by default they do not apply to that account if accessed through an **`ssh`** login from a remote system or a text-based login on a virtual console (such as **`tty5`**).

NOTE

You can make your shell environment use the same **`LANG`** setting as your graphical environment, even when you log in through a text-based virtual console or over **`ssh`**. One way to do this is to place code similar to the following in your **`~/.bashrc`** file. This example code will set the language used on a text login to match the one currently set for the user's GNOME desktop environment:

```
i=$(grep 'Language=' /var/lib/AccountsService/users/${USER} \
  | sed 's/Language=//')
if [ "$i" != "" ]; then
    export LANG=$i
fi
```

Japanese, Korean, Chinese, and other languages with a non-Latin character set might not display properly on text-based virtual consoles.

Individual commands can be made to use another language by setting the `LANG` variable on the command line:

```
[user@host ~]$ LANG=fr_FR.utf8 date
jeu. avril 25 17:55:01 CET 2019
```

Subsequent commands will revert to using the system's default language for output. The **`locale`** command can be used to determine the current value of `LANG` and other related environment variables.

Input Method Settings

GNOME 3 in Red Hat Enterprise Linux 7 or later automatically uses the ***IBus*** input method selection system, which makes it easy to change keyboard layouts and input methods quickly.

The ***Region & Language*** application can also be used to enable alternative input methods. In the ***Region & Language*** application window, the ***Input Sources*** box shows what input methods are currently available. By default, ***English (US)*** may be the only available method. Highlight ***English (US)*** and click the ***keyboard*** icon to see the current keyboard layout.

To add another input method, click the ***+*** button at the bottom left of the ***Input Sources*** window. An ***Add an Input Source*** window will open. Select your language, and then your preferred input method or keyboard layout.

When more than one input method is configured, the user can switch between them quickly by typing **`Super`**+**`Space`** (sometimes called **`Windows`**+**`Space`**). A *status indicator* will also appear in the GNOME top bar, which has two functions: It indicates which input method is active, and acts as a menu that can be used to switch between input methods or select advanced features of more complex input methods.

Some of the methods are marked with gears, which indicate that those methods have advanced configuration options and capabilities. For example, the Japanese ***Japanese (Kana Kanji)*** input method allows the user to pre-edit text in Latin and use **`Down Arrow`** and **`Up Arrow`** keys to select the correct characters to use.

US English speakers may also find this useful. For example, under ***English (United States)*** is the keyboard layout ***English (international AltGr dead keys)***, which treats **`AltGr`** (or the right **`Alt`**) on a PC 104/105-key keyboard as a "secondary shift" modifier key and dead key activation key for typing additional characters. There are also Dvorak and other alternative layouts available.

> **NOTE**
> Any Unicode character can be entered in the GNOME desktop environment if you know the character's Unicode code point. Type **`Ctrl`**+**`Shift`**+**`U`**, followed by the code point. After **`Ctrl`**+**`Shift`**+**`U`** has been typed, an underlined **`u`** will be displayed to indicate that the system is waiting for Unicode code point entry.
>
> For example, the lowercase Greek letter lambda has the code point U+03BB, and can be entered by typing **`Ctrl`**+**`Shift`**+**`U`**, then **`03BB`**, then **`Enter`**.

SYSTEM-WIDE DEFAULT LANGUAGE SETTINGS

The system's default language is set to US English, using the UTF-8 encoding of Unicode as its character set (**`en_US.utf8`**), but this can be changed during or after installation.

From the command line, the `root` user can change the system-wide locale settings with the **`localectl`** command. If **`localectl`** is run with no arguments, it displays the current system-wide locale settings.

To set the system-wide default language, run the command **`localectl set-locale LANG=`*`locale`***, where *locale* is the appropriate value for the `LANG` environment variable from the "Language Codes Reference" table in this chapter. The change will take effect for users on their next login, and is stored in **`/etc/locale.conf`**.

```
[root@host ~]# localectl set-locale LANG=fr_FR.utf8
```

In GNOME, an administrative user can change this setting from ***Region & Language*** by clicking the ***Login Screen*** button at the upper-right corner of the window. Changing the ***Language*** of the graphical login screen will also adjust the system-wide default language setting stored in the **`/etc/locale.conf`** configuration file.

IMPORTANT

Text-based virtual consoles such as **`tty4`** are more limited in the fonts they can display than terminals in a virtual console running a graphical environment, or pseudoterminals for **`ssh`** sessions. For example, Japanese, Korean, and Chinese characters may not display as expected on a text-based virtual console. For this reason, you should consider using English or another language with a Latin character set for the system-wide default.

Likewise, text-based virtual consoles are more limited in the input methods they support, and this is managed separately from the graphical desktop environment. The available global input settings can be configured through **`localectl`** for both text-based virtual consoles and the graphical environment. See the **`localectl`**(1) and **`vconsole.conf`**(5) man pages for more information.

LANGUAGE PACKS

Special RPM packages called *langpacks* install language packages that add support for specific languages. These langpacks use dependencies to automatically install additional RPM packages containing localizations, dictionaries, and translations for other software packages on your system.

To list the langpacks that are installed and that may be installed, use **`yum list langpacks-*`**:

```
[root@host ~]# yum list langpacks-*
Updating Subscription Management repositories.
Updating Subscription Management repositories.
Installed Packages
langpacks-en.noarch      1.0-12.el8        @AppStream
Available Packages
langpacks-af.noarch      1.0-12.el8        rhel-8-for-x86_64-appstream-rpms
langpacks-am.noarch      1.0-12.el8        rhel-8-for-x86_64-appstream-rpms
langpacks-ar.noarch      1.0-12.el8        rhel-8-for-x86_64-appstream-rpms
langpacks-as.noarch      1.0-12.el8        rhel-8-for-x86_64-appstream-rpms
langpacks-ast.noarch     1.0-12.el8        rhel-8-for-x86_64-appstream-rpms
```

To add language support, install the appropriate langpacks package. For example, the following command adds support for French:

```
[root@host ~]# yum install langpacks-fr
```

Use **`yum repoquery --whatsupplements`** to determine what RPM packages may be installed by a langpack:

```
[root@host ~]# yum repoquery --whatsupplements langpacks-fr
Updating Subscription Management repositories.
Updating Subscription Management repositories.
Last metadata expiration check: 0:01:33 ago on Wed 06 Feb 2019 10:47:24 AM CST.
glibc-langpack-fr-0:2.28-18.el8.x86_64
gnome-getting-started-docs-fr-0:3.28.2-1.el8.noarch
hunspell-fr-0:6.2-1.el8.noarch
hyphen-fr-0:3.0-1.el8.noarch
libreoffice-langpack-fr-1:6.0.6.1-9.el8.x86_64
man-pages-fr-0:3.70-16.el8.noarch
mythes-fr-0:2.3-10.el8.noarch
```

IMPORTANT

Langpacks packages use RPM *weak dependencies* in order to install supplementary packages only when the core package that needs it is also installed.

For example, when installing *langpacks-fr* as shown in the preceding examples, the *mythes-fr* package will only be installed if the *mythes* thesaurus is also installed on the system.

If *mythes* is subsequently installed on that system, the *mythes-fr* package will also automatically be installed due to the weak dependency from the already installed *langpacks-fr* package.

REFERENCES

`locale`(7), **`localectl`**(1), **`locale.conf`**(5), **`vconsole.conf`**(5), **`unicode`**(7), and **`utf-8`**(7) man pages

Conversions between the names of the graphical desktop environment's X11 layouts and their names in **`localectl`** can be found in the file **`/usr/share/X11/xkb/rules/base.lst`**.

LANGUAGE CODES REFERENCE

NOTE

This table might not reflect all langpacks available on your system. Use **`yum info langpacks-`*`SUFFIX`*** to get more information about any particular langpacks package.

Language Codes

LANGUAGE	LANGPACKS SUFFIX	$LANG VALUE
English (US)	en	en_US.utf8

LANGUAGE	LANGPACKS SUFFIX	$LANG VALUE
Assamese	as	as_IN.utf8
Bengali	bn	bn_IN.utf8
Chinese (Simplified)	zh_CN	zh_CN.utf8
Chinese (Traditional)	zh_TW	zh_TW.utf8
French	fr	fr_FR.utf8
German	de	de_DE.utf8
Gujarati	gu	gu_IN.utf8
Hindi	hi	hi_IN.utf8
Italian	it	it_IT.utf8
Japanese	ja	ja_JP.utf8
Kannada	kn	kn_IN.utf8
Korean	ko	ko_KR.utf8
Malayalam	ml	ml_IN.utf8
Marathi	mr	mr_IN.utf8
Odia	or	or_IN.utf8
Portuguese (Brazilian)	pt_BR	pt_BR.utf8
Punjabi	pa	pa_IN.utf8
Russian	ru	ru_RU.utf8
Spanish	es	es_ES.utf8
Tamil	ta	ta_IN.utf8
Telugu	te	te_IN.utf8

CHAPTER 1

IMPROVING COMMAND-LINE PRODUCTIVITY

GOAL

Run commands more efficiently by using advanced features of the Bash shell, shell scripts, and various utilities provided by Red Hat Enterprise Linux.

OBJECTIVES

- Automate sequences of commands by writing a simple shell script.
- Efficiently run commands over lists of items in a script or from the command-line using for loops and conditionals.
- Find text matching a pattern in log files and command output using the **`grep`** command and regular expressions.

SECTIONS

- Writing Simple Bash Scripts (and Guided Exercise)
- Running Commands More Efficiently Using Loops (and Guided Exercise)
- Matching Text in Command Output with Regular Expressions (and Guided Exercise)

LAB

Improving Command-line Productivity

WRITING SIMPLE BASH SCRIPTS

OBJECTIVES

After completing this section, you should be able to automate sequences of commands by writing a simple shell script.

CREATING AND EXECUTING BASH SHELL SCRIPTS

Many simple, common system administration tasks are accomplished using command-line tools. Tasks with greater complexity often require chaining together multiple commands that pass results between them. Using the Bash shell environment and scripting features, Linux commands are combined into *shell scripts* to easily solve repetitive and difficult real-world problems.

In its simplest form, a Bash shell script is an executable file that contains a list of commands, and possibly with programming logic to control decision-making in the overall task. When well-written, a shell script is a powerful command-line tool on its own, and can be leveraged by other scripts.

Shell scripting proficiency is essential to successful system administration in any operational environment. Working knowledge of shell scripting is crucial in enterprise environments, where script use can improve the efficiency and accuracy of routine task completion.

You can create a Bash shell script by opening a new empty file in a text editor. While you can use any text editor, advanced editors, such as **`vim`** or **`emacs`**, understand Bash shell syntax and can provide **`color-coded`** highlighting. This highlighting helps identify common errors such as improper syntax, unpaired quotes, unclosed parenthesis, braces, and brackets, and much more.

Specifying the Command Interpreter

The first line of a script begins with the notation '**`#!`**', commonly referred to as **`sh-bang`** or **`she-bang`**, from the names of those two characters, **`sharp`** and **`bang`**. This specific two-byte **`magic number`** notation indicates an interpretive script; syntax that follows the notation is the fully-qualified filename for the correct **`command interpreter`** needed to process this script's lines. To understand how **`magic numbers`** indicate file types in Linux, see the **`file(1)`** and **`magic(5)`** man pages. For script files using Bash scripting syntax, the first line of a shell script begins as follows:

```
#!/bin/bash
```

Executing a Bash Shell Script

A completed shell script must be executable to run as an ordinary command. Use the **`chmod`** command to add execute permission, possibly in conjunction with the **`chown`** command to change the file ownership of the script. Grant execute permission only for intended users of the script.

If you place the script in one of the directories listed in the shell's `PATH` environmental variable, then you can invoke the shell script using the file name alone as with any other command. The shell uses the first command it finds with that file name; avoid using existing command names for your shell script file name. Alternatively, you can invoke a shell script by entering a path name to the

script on the command line. The **which** command, followed by the file name of the executable script, displays the path name to the command that will be executed.

```
[user@host ~]$ which hello
~/bin/hello
[user@host ~]$ echo $PATH
/home/user/.local/bin:/home/user/bin:/usr/share/Modules/bin:/usr/local/bin:/usr/
bin:/usr/local/sbin:/usr/sbin
```

Quoting Special Characters

A number of characters and words have special meaning to the Bash shell. However, occasionally you will want to use these characters for their literal values, rather than for their special meanings. To do this, use one of three tools to remove (or *escape*) the special meaning: the backslash (\), single quotes ('), or double quotes ("").

The backslash escape character removes the special meaning of the single character immediately following it. For example, to display the literal string **# not a comment** with the **echo** command, the # sign must not be interpreted by Bash as having special meaning. Place the backslash character in front of the # sign.

```
[user@host ~]$ echo # not a comment

[user@host ~]$ echo \# not a comment
# not a comment
```

When you need to escape more than one character in a text string, either use the escape character multiple times or employ single quotes ('). Single quotes preserve the literal meaning of all characters they enclose. Observe the escape character and single quotes in action:

```
[user@host ~]$ echo # not a comment #

[user@host ~]$ echo \# not a comment #
# not a comment
[user@host ~]$ echo \# not a comment \#
# not a comment #
[user@host ~]$ echo '# not a comment #'
# not a comment #
```

Use double quotation marks to suppress globbing and shell expansion, but still allow command and variable substitution. Variable substitution is conceptually identical to command substitution, but may use optional brace syntax. Observe the examples of various forms of quotation mark use below.

Use single quotation marks to interpret all text literally. Besides suppressing globbing and shell expansion, quotations direct the shell to additionally suppress command and variable substitution. The question mark (?) is a **meta-character** that also needs protection from expansion.

```
[user@host ~]$ var=$(hostname -s); echo $var
host
[user@host ~]$ echo "***** hostname is ${var} *****"
***** hostname is host *****
[user@host ~]$ echo Your username variable is \$USER.
```

```
Your username variable is $USER.
[user@host ~]$ echo "Will variable $var evaluate to $(hostname -s)?"
Will variable host evaluate to host?
[user@host ~]$ echo 'Will variable $var evaluate to $(hostname -s)?'
Will variable $var evaluate to $(hostname -s)?
[user@host ~]$ echo "\"Hello, world\""
"Hello, world"
[user@host ~]$ echo '"Hello, world"'
"Hello, world"
```

Providing Output From a Shell Script

The **echo** command displays arbitrary text by passing the text as an argument to the command. By default, the text displays on *standard output (STDOUT)*, but it can also be directed to *standard error (STDERR)* using output redirection. In the following simple Bash script, the **echo** command displays the message "**Hello, world**" to STDOUT.

```
[user@host ~]$ cat ~/bin/hello
#!/bin/bash

echo "Hello, world"

[user@host ~]$ hello
Hello, world
```

NOTE

This user can just run **hello** at the prompt because the **~/bin** (**/home/user/bin**) directory is in the user's PATH variable and the **hello** script in it is executable. The shell automatically finds the script there, as long as there is no other executable file called **hello** in any of the directories listed prior to **/home/user/bin** in the PATH.

The **echo** command is widely used in shell scripts to display informational or error messages. These messages can be a helpful indicator of the progress of a script and can be directed either to standard output, standard error, or be redirected to a log file for archiving. When displaying error messages, it is good practice to direct them to STDERR to make it easier to differentiate error messages from normal status messages.

```
[user@host ~]$ cat ~/bin/hello
#!/bin/bash

echo "Hello, world"
echo "ERROR: Houston, we have a problem." >&2

[user@host ~]$ hello 2> hello.log
Hello, world
[user@host ~]$ cat hello.log
ERROR: Houston, we have a problem.
```

The **echo** command can also be very helpful when trying to debug a problematic shell script. The addition of **echo** statements to the portion of the script that is not behaving as expected can help clarify the commands being executed, as well as the values of variables being invoked.

REFERENCES

bash(1), **magic**(5), **echo**(1), and **echo**(1p), man pages.

GUIDED EXERCISE

WRITING SIMPLE BASH SCRIPTS

In this exercise, you will write a simple Bash script containing a sequence of commands and run it from the command line.

OUTCOMES

You should be able to:

- Write and execute a simple Bash script.
- Redirect the output of a simple Bash script to a file.

BEFORE YOU BEGIN

Log in to `workstation` as `student` using `student` as the password.

On `workstation`, run the **`lab console-write start`** command. This command runs a start script that determines if the `servera` machine is reachable on the network. The script will alert you if it is not available. It also installs the *vim-enhanced* package if needed.

```
[student@workstation ~]$ lab console-write start
```

1. From `workstation`, open an SSH session to `servera` as `student`. The systems are configured to use SSH keys for authentication, therefore a password is not required.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

2. Create and execute a simple Bash script.

   2.1. Use the **`vim`** text editor to create a new text file under your home directory, and name it **`firstscript.sh`**

```
[student@servera ~]$ vim firstscript.sh
```

   2.2. Insert the following text, and save the file. Note that the number of hash signs (#) is arbitrary.

```
#!/bin/bash
echo "This is my first bash script" > ~/output.txt
echo "" >> ~/output.txt
echo "#######################################################" >> ~/
output.txt
```

   2.3. Use the **sh** command to execute the script.

```
[student@servera ~]$ sh firstscript.sh
```

2.4. Review the output file generated by the script.

```
[student@servera ~]$ cat output.txt
This is my first bash script

#####################################################
```

3. Add more commands to the **`firstscript.sh`** script, execute it, and review the output.

3.1. Use the **`vim`** text editor to edit **`firstscript.sh`**

```
[student@servera ~]$ vim firstscript.sh
```

3.2. Append the following lines in bold to the **`firstscript.sh`** file.

```
#!/bin/bash
#
echo "This is my first bash script" > ~/output.txt
echo "" >> ~/output.txt
echo "#####################################################" >> ~/
output.txt
echo "LIST BLOCK DEVICES" >> ~/output.txt
echo "" >> ~/output.txt
lsblk >> ~/output.txt
echo "" >> ~/output.txt
echo "#####################################################" >> ~/
output.txt
echo "FILESYSTEM FREE SPACE STATUS" >> ~/output.txt
echo "" >> ~/output.txt
df -h >> ~/output.txt
echo "#####################################################" >> ~/
output.txt
```

3.3. Make the **`firstscript.sh`** file executable using the **`chmod`** command.

```
[student@servera ~]$ chmod a+x firstscript.sh
```

3.4. Execute the **`firstscript.sh`** script.

```
[student@servera ~]$ ./firstscript.sh
```

3.5. Review the output file generated by the script.

```
[student@servera ~]$ cat output.txt
This is my first bash script

#####################################################
```

```
LIST BLOCK DEVICES

NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sr0     11:0    1 1024M  0 rom
vda    252:0    0   10G  0 disk
└─vda1 252:1    0   10G  0 part /
vdb    252:16   0    5G  0 disk

#####################################################
FILESYSTEM FREE SPACE STATUS

Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        892M     0  892M   0% /dev
tmpfs           915M     0  915M   0% /dev/shm
tmpfs           915M   17M  899M   2% /run
tmpfs           915M     0  915M   0% /sys/fs/cgroup
/dev/vda1        10G  1.5G  8.6G  15% /
tmpfs           183M     0  183M   0% /run/user/1000
#####################################################
```

4. Remove the exercise files and log off from `servera`.

   4.1. Delete the script file **`firstscript.sh`** and output file **`output.txt`**.

```
[student@servera ~]$ rm firstscript.sh output.txt
```

   4.2. Log off from `servera`.

```
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```

Finish

On `workstation`, run the **`lab console-write finish`** script to complete this exercise.

```
[student@workstation ~]$ lab console-write finish
```

This concludes the guided exercise.

RUNNING COMMANDS MORE EFFICIENTLY USING LOOPS

OBJECTIVES

After completing this section, you should be able to:

- Iterate over lists using **`for`** loops.
- Evaluate exit codes from commands and scripts.
- Perform tests using operators.
- Create conditional structures using **`if`** statements.

USING LOOPS TO ITERATE COMMANDS

System administrators often encounter repetitive tasks in their day-to-day activities. Repetitive tasks can take the form of executing an action multiple times on a target, such as checking a process every minute for 10 minutes to see if it has completed. Task repetition can also take the form of executing an action once across multiple targets, such as backing up each database on a system. The *for loop* is one of the multiple shell looping constructs offered by Bash, and can be used for task iterations.

Processing Items from the Command Line

Bash's for loop construct uses the following syntax.

```
for VARIABLE in LIST; do
COMMAND VARIABLE
done
```

The loop processes the strings provided in *LIST* in order one by one and exits after processing the last string in the list. Each string in the list is temporarily stored as the value of *VARIABLE*, while the **`for`** loop executes the block of commands contained in its construct. The naming of the variable is arbitrary. Typically, the variable value is referenced by commands in the command block.

The list of strings provided to a **`for`** loop can be supplied in several ways. It can be a list of strings entered directly by the user, or be generated from different types of shell expansion, such as variable, brace, or file-name expansion, or command substitution. Some examples that demonstrate the different ways strings can be provided to **`for`** loops follow.

```
[user@host ~]$ for HOST in host1 host2 host3; do echo $HOST; done
host1
host2
host3
[user@host ~]$ for HOST in host{1,2,3}; do echo $HOST; done
host1
host2
host3
[user@host ~]$ for HOST in host{1..3}; do echo $HOST; done
host1
```

```
host2
host3
[user@host ~]$ for FILE in file*; do ls $FILE; done
filea
fileb
filec
[user@host ~]$ for FILE in file{a..c}; do ls $FILE; done
filea
fileb
filec
[user@host ~]$ for PACKAGE in $(rpm -qa | grep kernel); \
do echo "$PACKAGE was installed on \
$(date -d @$(rpm -q --qf "%{INSTALLTIME}\n" $PACKAGE))"; done
abrt-addon-kerneloops-2.1.11-12.el7.x86_64 was installed on Tue Apr 22 00:09:07
 EDT 2014
kernel-3.10.0-121.el7.x86_64 was installed on Thu Apr 10 15:27:52 EDT 2014
kernel-tools-3.10.0-121.el7.x86_64 was installed on Thu Apr 10 15:28:01 EDT 2014
kernel-tools-libs-3.10.0-121.el7.x86_64 was installed on Thu Apr 10 15:26:22 EDT
 2014
[user@host ~]$ for EVEN in $(seq 2 2 10); do echo "$EVEN"; done
2
4
6
8
10
```

USING EXIT CODES WITHIN A SCRIPT

After a script has processed all of its contents, it exits to the process that called it. However, there may be times when it is desirable to exit a script before it finishes, such as when an error condition is encountered. This can be accomplished with the use of the **exit** command within a script. When a script encounters the **exit** command, it exits immediately and does not process the remainder of the script.

The **exit** command can be executed with an optional integer argument between **0** and **255**, which represents an exit code. An exit code is a code that is returned after a process has completed. An exit code value of **0** represents no error. All other nonzero values indicate an error exit code. You can use different nonzero values to differentiate between different types of errors encountered. This exit code is passed back to the parent process, which stores it in the ? variable and can be accessed with **$?** as demonstrated in the following examples.

```
[user@host bin]$ cat hello
#!/bin/bash
echo "Hello, world"
exit 0

[user@host bin]$ ./hello
Hello, world

[user@host bin]$ echo $?
0
```

If the **exit** command is called without an argument, then the script exits and passes the exit status of the last command executed to the parent process.

TESTING SCRIPT INPUTS

To ensure that scripts are not easily disrupted by unexpected conditions, it is good practice to not make assumptions regarding input, such as command-line arguments, user input, command substitutions, variable expansions, and file-name expansions. Integrity checking can be performed by using Bash's **`test`** command.

Like all commands, the **`test`** command produces an exit code upon completion, which is stored as the value **`$?`**. To see the conclusion of a test, display the value of **`$?`** immediately following the execution of the **`test`** command. Again, an exit status value of **`0`** indicates the test succeeded, and nonzero values indicate the test failed.

Tests are performed using a variety of *operators*. Operators can be used to determine if a number is greater than, greater than or equal to, less than, less than or equal to, or equal to another number. They can be used to test if a string of text is the same or not the same as another string of text. Operators can also be used to evaluate if a variable has a value or not.

NOTE

Many types of operators are used in shell scripting, in addition to the comparison operators taught here. The man page for **`test(1)`** lists the important conditional expression operators with descriptions. The **`bash(1)`** man page also explains operator use and evaluation, but is a very difficult read for beginners. It is recommended that students further their advanced shell scripting needs through books and courses dedicated to shell programming.

The following examples demonstrate the use of the **`test`** command using Bash's numeric comparison operators.

```
[user@host ~]$ test 1 -gt 0 ; echo $?
0
[user@host ~]$ test 0 -gt 1 ; echo $?
1
```

Tests can be performed using the Bash test command syntax, **`[ <TESTEXPRESSION> ]`**. They can also be performed using Bash's newer extended test command syntax, **`[[ <TESTEXPRESSION> ]]`**, which has been available since Bash version 2.02 and provides features such as glob pattern matching and regex pattern matching.

The following examples demonstrate the use of Bash's test command syntax and Bash's numeric comparison operators.

```
[user@host ~]$ [ 1 -eq 1 ]; echo $?
0
[user@host ~]$ [ 1 -ne 1 ]; echo $?
1
[user@host ~]$ [ 8 -gt 2 ]; echo $?
0
[user@host ~]$ [ 2 -ge 2 ]; echo $?
0
[user@host ~]$ [ 2 -lt 2 ]; echo $?
```

```
1
[user@host ~]$ [ 1 -lt 2 ]; echo $?
0
```

The following examples demonstrate the use of Bash's string comparison operators.

```
[user@host ~]$ [ abc = abc ]; echo $?
0
[user@host ~]$ [ abc == def ]; echo $?
1
[user@host ~]$ [ abc != def ]; echo $?
0
```

The following examples demonstrate the use of Bash's string unary operators.

```
[user@host ~]$ STRING=''; [ -z "$STRING" ]; echo $?
0
[user@host ~]$ STRING='abc'; [ -n "$STRING" ]; echo $?
0
```

NOTE

The space characters inside the test brackets are mandatory, because they separate the words and elements within the test expression. The shell's command parsing routine divides all command lines into words and operators by recognizing spaces and other metacharacters, using built-in parsing rules. For full treatment of this advanced concept, see the man page **`getopt(3)`**. The left square bracket character ([) is itself a built-in alias for the **`test`** command. Shell words, whether they are commands, subcommands, options, arguments or other token elements, are always delimited by spaces.

CONDITIONAL STRUCTURES

Simple shell scripts represent a collection of commands that are executed from beginning to end. Conditional structures allow users to incorporate decision making into shell scripts, so that certain portions of the script are executed only when certain conditions are met.

Using the if/then Construct

The simplest of the conditional structures in Bash is the if/then construct, which has the following syntax.

```
if <CONDITION>; then
      <STATEMENT>
      ...
      <STATEMENT>
fi
```

With this construct, if a given condition is met, one or more actions are taken. If the given condition is not met, then no action is taken. The numeric, string, and file tests previously demonstrated are frequently utilized for testing the conditions in **`if/then`** statements. The **`fi`**

statement at the end closes the **`if/then`** construct. The following code section demonstrates the use of an **`if/then`** construct to start the `psacct` service if it is not active.

```
[user@host ~]$ systemctl is-active psacct > /dev/null 2>&1
[user@host ~]$ if  [ $? -ne 0 ]; then
> sudo systemctl start psacct
> fi
```

Using the if/then/else Construct

The **`if/then`** construct can be further expanded so that different sets of actions can be taken depending on whether a condition is met. This is accomplished with the **`if/then/else`** construct.

```
if <CONDITION>; then
      <STATEMENT>
      ...
      <STATEMENT>
    else
      <STATEMENT>
      ...
      <STATEMENT>
fi
```

The following code section demonstrates the use of an **`if/then/else`** statement to start the `psacct` service if it is not active and to stop it if it is active.

```
[user@host ~]$ systemctl is-active psacct > /dev/null 2>&1
[user@host ~]$ if  [ $? -ne 0 ]; then
> sudo systemctl start psacct
> else
> sudo systemctl stop psacct
> fi
```

Using the `if/then/elif/then/else` Construct

Lastly, the **`if/then/else`** construct can be further expanded to test more than one condition, executing a different set of actions when a condition is met. The construct for this is shown in the following example:

```
if <CONDITION>; then
      <STATEMENT>
      ...
      <STATEMENT>
    elif <CONDITION>; then
      <STATEMENT>
      ...
      <STATEMENT>
    else
      <STATEMENT>
      ...
      <STATEMENT>
    fi
```

In this conditional structure, Bash tests the conditions in the order presented. When it finds a condition that is true, Bash executes the actions associated with the condition and then skips the remainder of the conditional structure. If none of the conditions are true, Bash executes the actions enumerated in the **else** clause.

The following code section demonstrates the use of an **if/then/elif/then/else** statement to run the **mysql** client if the `mariadb` service is active, run the **psql** client if the `postgresql` service is active, or run the **sqlite3** client if both the `mariadb` and `postgresql` services are not active.

```
[user@host ~]$ systemctl is-active mariadb > /dev/null 2>&1
MARIADB_ACTIVE=$?
[user@host ~]$ sudo systemctl is-active postgresql > /dev/null 2>&1
POSTGRESQL_ACTIVE=$?
[user@host ~]$ if  [ "$MARIADB_ACTIVE" -eq 0 ]; then
> mysql
> elif  [ "$POSTGRESQL_ACTIVE" -eq 0 ]; then
> psql
> else
> sqlite3
> fi
```

REFERENCES

bash(1) man page

GUIDED EXERCISE

RUNNING COMMANDS MORE EFFICIENTLY USING LOOPS

In this exercise, you will use loops to efficiently print the host name from multiple servers.

OUTCOMES

You should be able to create a **for** loop to iterate through a list of items from the command-line and in a shell script.

BEFORE YOU BEGIN

Log in as the `student` user on `workstation` using `student` as the password.

On `workstation`, run the **`lab console-commands start`** command. The command runs a start script that determines if the `servera` and `serverb` hosts are reachable on the network. The script will alert you if they are not available.

```
[student@workstation ~]$ lab console-commands start
```

1. Use the **ssh** and **hostname** commands to print the host name of `servera` and `serverb` to standard output.

```
[student@workstation ~]$ ssh student@servera hostname
servera.lab.example.com
[student@workstation ~]$ ssh student@serverb hostname
serverb.lab.example.com
```

2. Create a **for** loop to perform the same task more efficiently.

```
[student@workstation ~]$ for HOST in servera serverb
do
ssh student@${HOST} hostname
done
servera.lab.example.com
serverb.lab.example.com
```

3. Create a shell script to execute the same **for** loop.

   3.1. Create the **/home/student/bin** directory to contain the shell script.

```
[student@workstation ~]$ mkdir ~/bin
```

   3.2. Verify that the newly created directory is in your PATH environmental variable.

```
[student@workstation ~]$ echo $PATH
/home/student/.local/bin:/home/student/bin::/usr/local/bin:/usr/bin:/
usr/local/sbin:/usr/sbin
```

3.3. Create a shell script at **/home/student/bin/printhostname.sh** to perform the **for** loop. Use the **cat** command to verify the content of **printhostname.sh**.

```
[student@workstation ~]$ vim ~/bin/printhostname.sh
[student@workstation ~]$ cat ~/bin/printhostname.sh
#!/bin/bash
#Execute for loop to print server hostname.
for HOST in servera serverb
do
  ssh student@${HOST} hostname
done
exit 0
```

3.4. Ensure the newly created script is executable.

```
[student@workstation ~]$ chmod +x ~/bin/printhostname.sh
```

3.5. Run the script from your home directory.

```
[student@workstation ~]$ printhostname.sh
servera.lab.example.com
serverb.lab.example.com
```

3.6. Verify that the exit code of your script is 0.

```
[student@workstation ~]$ echo $?
0
```

Finish

On `workstation`, run the **lab console-commands finish** script to finish this exercise.

```
[student@workstation ~]$ lab console-commands finish
```

This concludes the guided exercise.

MATCHING TEXT IN COMMAND OUTPUT WITH REGULAR EXPRESSIONS

OBJECTIVES

After completing this section, students should be able to:

- Create regular expressions that match desired data.
- Apply regular expressions to text files using the **grep** command.
- Search files and data from piped commands using **grep**.

WRITING REGULAR EXPRESSIONS

Figure 1.0: Regular expression fundamentals

Regular expressions provide a pattern matching mechanism that facilitates finding specific content. The **vim** , **grep**, and **less** commands can all use regular expressions. Programming languages such as Perl, Python, and C can all use regular expressions when using pattern matching criteria.

Regular expressions are a language of their own, which means they have their own syntax and rules. This section looks at the syntax used when creating regular expressions, as well as showing some regular expression examples.

Describing a Simple Regular Expression

The simplest regular expression is an exact match. An exact match is when the characters in the regular expression match the type and order in the data that is being searched.

Suppose a user is looking through the following file looking for all occurrences of the pattern **cat**:

```
cat
dog
concatenate
dogma
category
educated
boondoggle
vindication
chilidog
```

cat is an exact match of a **c**, followed by an **a**, followed by a **t** with no other characters in between. Using **cat** as the regular expression to search the previous file returns the following matches:

```
cat
concatenate
category
educated
vindication
```

Matching the Start and End of a Line

The previous section used an exact match regular expression on a file. Note that the regular expression would match the search string no matter where on the line it occurred: the beginning, end, or middle of the word or line. Use a *line anchor* to control the location of where the regular expression looks for a match.

To search at the beginning of a line, use the caret character (^). To search at the end of a line, use the dollar sign ($).

Using the same file as above, the **`^cat`** regular expression would match two words. The **`$cat`** regular expression would not find any matching words.

```
cat
dog
concatenate
dogma
category
educated
boondoggle
vindication
chilidog
```

To locate lines in the file ending with **`dog`**, use that exact expression and an end of line anchor to create the regular expression **`dog$`**. Applying **`dog$`** to the file would find two matches:

```
dog
chilidog
```

To locate the only word on a line, use both the beginning and end-of-line anchors. For example, to locate the word **`cat`** when it is the only word on a line, use **`^cat$`**.

```
cat dog rabbit
cat
horse cat cow
cat pig
```

Adding Wildcards and Multipliers to Regular Expressions

Regular expressions use a period or dot (.) to match any single character with the exception of the newline character. A regular expression of **`c.t`** searches for a string that contains a **`c`** followed by any single character followed by a **`t`**. Example matches include **`cat`**, **`concatenate`**, **`vindication`**, **`c5t`**, and **`c$t`**.

Using an unrestricted wildcard you cannot predict the character that would match the wildcard. To match specific characters, replace the unrestricted wildcard with acceptable characters. Changing

the regular expression to **`c[aou]t`** matches patterns that start with a **`c`**, followed by either an **`a`**, **`o`**, or **`u`**, followed by a **`t`**.

Multipliers are a mechanism used often with wildcards. Multipliers apply to the previous character in the regular expression. One of the more common multipliers used is the asterisk, or star character (*). When used in a regular expression, this multiplier means match zero or more of the previous expression. You can use * with expressions, not just characters. For example, **`c[aou]*t`**. A regular expression of **`c.*t`** matches **`cat`**, **`coat`**, **`culvert`**, and even **`ct`** (zero characters between the **`c`** and the **`t`**). Any data starting with a **`c`**, then zero or more characters, ending with a **`t`**.

Another type of multiplier would indicate the number of previous characters desired in the pattern. An example of using an explicit multiplier would be **`'c.\{2\}t'`**. This regular expression will match any word beginning with a **`c`**, followed by exactly any two characters, and ending with a **`t`**. **`'c.\{2\}t'`** would match two words in the example below:

```
cat
coat convert
cart covert
cypher
```

NOTE

It is recommend practice to use single quotes to encapsulate the regular expression because they often contain shell metacharacters (such as **`$`**, **`*`**, and **`{}`**). This ensures that the characters are interpreted by the command and not by the shell.

NOTE

This course has introduced two distinct metacharacter text parsing systems: shell *pattern matching* (also known as file globbing or file-name expansion), and *regular expressions*. Because both systems use similar metacharacters, such as the asterisk (*), but have differences in metacharacter interpretation and rules, the two systems can be confusing until each is sufficiently mastered.

Pattern matching is a command-line parsing technique designed for specifying many file names easily, and is primarily supported only for representing file-name patterns on the command line. Regular expressions are designed to represent any form or pattern in text strings, no matter how complex. Regular expression are internally supported by numerous text processing commands, such as **`grep`**, **`sed`**, **`awk`**, **`python`**, **`perl`**, and many applications, with some minimal command-dependent variations in interpretation rules.

Regular Expressions

OPTION	DESCRIPTION
.	The period (.) matches any single character.
?	The preceding item is optional and will be matched at most once.
*****	The preceding item will be matched zero or more times.
+	The preceding item will be matched one or more times.

OPTION	DESCRIPTION
{n}	The preceding item is matched exactly n times.
{n,}	The preceding item is matched n or more times.
{,m}	The preceding item is matched at most m times.
{n,m}	The preceding item is matched at least n times, but not more than m times.
[:alnum:]	Alphanumeric characters: '[:alpha:]' and '[:digit:]'; in the 'C' locale and ASCII character encoding, this is the same as '[0-9A-Za-z]'.
[:alpha:]	Alphabetic characters: '[:lower:]' and '[:upper:]'; in the 'C' locale and ASCII character encoding, this is the same as '[A-Za-z]'.
[:blank:]	Blank characters: space and tab.
[:cntrl:]	Control characters. In ASCII, these characters have octal codes 000 through 037, and 177 (DEL). In other character sets, these are the equivalent characters, if any.
[:digit:]	Digits: 0 1 2 3 4 5 6 7 8 9.
[:graph:]	Graphical characters: '[:alnum:]' and '[:punct:]'.
[:lower:]	Lower-case letters; in the 'C' locale and ASCII character encoding, this is a b c d e f g h i j k l m n o p q r s t u v w x y z.
[:print:]	Printable characters: '[:alnum:]', '[:punct:]', and space.
[:punct:]	Punctuation characters; in the 'C' locale and ASCII character encoding, this is! " # $ % & ' ( ) * + , - . / : ; < = > ? @ [\ ] ^ _ ' { | } ~. In other character sets, these are the equivalent characters, if any.
[:space:]	Space characters: in the 'C' locale, this is tab, newline, vertical tab, form feed,carriage return, and space.
[:upper:]	Upper-case letters: in the 'C' locale and ASCII character encoding, this is A B C D E F G H I J K L M N O P Q R S T U V W X Y Z.
[:xdigit:]	Hexadecimal digits: 0 1 2 3 4 5 6 7 8 9 A B C D E F a b c d e f.
\b	Match the empty string at the edge of a word.
\B	Match the empty string provided it is not at the edge of a word.
\<	Match the empty string at the beginning of word.
\>	Match the empty string at the end of word.
\w	Match word constituent. Synonym for '[_[:alnum:]]'.
\W	Match non-word constituent. Synonym for '[^_[:alnum:]]'.
\s	Match white space. Synonym for '[[:space:]]'.

OPTION	DESCRIPTION
\S	Match non-whitespace. Synonym for '[^[:space:]]'.

MATCHING REGULAR EXPRESSIONS WITH GREP

The **grep** command, provided as part of the distribution, uses regular expressions to isolate matching data.

Isolating data using the grep command

The **grep** command provides a regular expression and a file on which the regular expression should be matched.

```
[user@host ~]$ grep '^computer' /usr/share/dict/words
computer
computerese
computerise
computerite
computerizable
computerization
computerize
computerized
computerizes
computerizing
computerlike
computernik
computers
```

NOTE
It is recommend practice to use single quotes to encapsulate the regular expression because they often contain shell metacharacters (such as $, *, and {}). This ensures that the characters are interpreted by **grep** and not by the shell.

The **grep** command can be used in conjunction with other commands using a pipe operator (|). For example:

```
[root@host ~]# ps aux | grep chrony
chrony     662  0.0  0.1  29440  2468 ?        S    10:56   0:00 /usr/sbin/chronyd
```

grep Options

The **grep** command has many useful options for adjusting how it uses the provided regular expression with data.

Table of Common `grep` Options

OPTION	FUNCTION
`-i`	Use the regular expression provided but do not enforce case sensitivity (run case-insensitive).
`-v`	Only display lines that do *not* contain matches to the regular expression.
`-r`	Apply the search for data matching the regular expression recursively to a group of files or directories.
`-A` *`NUMBER`*	Display *NUMBER* of lines after the regular expression match.
`-B` *`NUMBER`*	Display *NUMBER* of lines before the regular expression match.
`-e`	With multiple **`-e`** options used, multiple regular expressions can be supplied and will be used with a logical OR.

There are many other options to **`grep`**. Use the **`man`** page to research them.

`grep` Examples

The next examples use varied configuration files and log files.

Regular expressions are case-sensitive by default. Use the **`-i`** option with **`grep`** to run a case-insensitive search. The following example searches for the pattern **`serverroot`**.

```
[user@host ~]$ cat /etc/httpd/conf/httpd.conf
...output omitted...
ServerRoot "/etc/httpd"

#
# Listen: Allows you to bind Apache to specific IP addresses and/or
# ports, instead of the default. See also the <VirtualHost>
# directive.
#
# Change this to Listen on specific IP addresses as shown below to
# prevent Apache from glomming onto all bound IP addresses.
#
#Listen 12.34.56.78:80
Listen 80
```

```
[user@host ~]$ grep -i serverroot /etc/httpd/conf/httpd.conf
# with "/", the value of ServerRoot is prepended -- so 'log/access_log'
# with ServerRoot set to '/www' will be interpreted by the
# ServerRoot: The top of the directory tree under which the server's
# ServerRoot at a non-local disk, be sure to specify a local disk on the
# same ServerRoot for multiple httpd daemons, you will need to change at
ServerRoot "/etc/httpd"
```

In cases where you know what you are *not* looking for, the **-v** option is very useful. The **-v** option only displays lines that do not match the regular expression. In the following example, all lines, regardless of case, that do not contain the regular expression **server** are returned.

```
[user@host ~]$ cat /etc/hosts
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6

172.25.254.254  classroom.example.com classroom
172.25.254.254  content.example.com content
172.25.254.254  materials.example.com materials
172.25.250.254  workstation.lab.example.com workstation
### rht-vm-hosts file listing the entries to be appended to /etc/hosts

172.25.250.10   servera.lab.example.com servera
172.25.250.11   serverb.lab.example.com serverb
172.25.250.254  workstation.lab.example.com workstation
```

```
[user@host ~]$ grep -v -i server /etc/hosts
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6

172.25.254.254  classroom.example.com classroom
172.25.254.254  content.example.com content
172.25.254.254  materials.example.com materials
172.25.250.254  workstation.lab.example.com workstation
### rht-vm-hosts file listing the entries to be appended to /etc/hosts

172.25.250.254  workstation.lab.example.com workstation
```

To look at a file without being distracted by comment lines use the **-v** option. In the following example, the regular expression matches all lines that begin with a # or ; (typical characters that indicate the line will be interpreted as a comment). Those lines are then omitted from the output.

```
[user@host ~]$ cat /etc/ethertypes
#
# Ethernet frame types
#       This file describes some of the various Ethernet
#       protocol types that are used on Ethernet networks.
#
# This list could be found on:
#         http://www.iana.org/assignments/ethernet-numbers
#         http://www.iana.org/assignments/ieee-802-numbers
#
# <name>    <hexnumber> <alias1>...<alias35> #Comment
#
IPv4        0800    ip ip4      # Internet IP (IPv4)
X25     0805
ARP     0806    ether-arp   #
FR_ARP      0808            # Frame Relay ARP        [RFC1701]
```

```
[user@host ~]$ grep -v '^[#;]' /etc/ethertypes
IPv4        0800    ip ip4      # Internet IP (IPv4)
X25     0805
ARP     0806    ether-arp   #
FR_ARP      0808            # Frame Relay ARP        [RFC1701]
```

The **grep** command with the **-e** option allows you to search for more than one regular expression at a time. The following example, using a combination of **less** and **grep**, locates all occurrences of **pam_unix**, **user root** and **Accepted publickey** in the **/var/log/secure** log file.

```
[root@host ~]# cat /var/log/secure | grep -e 'pam_unix' \
-e 'user root' -e 'Accepted publickey' | less
Mar 19 08:04:46 jegui sshd[6141]: pam_unix(sshd:session): session opened for user
 root by (uid=0)
Mar 19 08:04:50 jegui sshd[6144]: Disconnected from user root 172.25.250.254 port
 41170
Mar 19 08:04:50 jegui sshd[6141]: pam_unix(sshd:session): session closed for user
 root
Mar 19 08:04:53 jegui sshd[6168]: Accepted publickey for student from
 172.25.250.254 port 41172 ssh2: RSA SHA256:M8ikhcEDm2tQ95Z0o7ZvufqEixCFCt
+wowZLNzNlBT0
```

To search for text in a file opened using **vim** or **less**, use the slash character (/) and type the pattern to find. Press **Enter** to start the search. Press **N** to find the next match.

```
[root@host ~]# vim /var/log/boot.log
...output omitted...
[^[[0;32m  OK  ^[[0m] Reached target Initrd Default Target.^M
  Starting dracut pre-pivot and cleanup hook...^M
[^[[0;32m  OK  ^[[0m] Started dracut pre-pivot and cleanup hook.^M
  Starting Cleaning Up and Shutting Down Daemons...^M
  Starting Plymouth switch root service...^M
  Starting Setup Virtual Console...^M
[^[[0;32m  OK  ^[[0m] Stopped target Timers.^M
[^[[0;32m  OK  ^[[0m] Stopped dracut pre-pivot and cleanup hook.^M
[^[[0;32m  OK  ^[[0m] Stopped target Initrd Default Target.^M
```

```
[root@host ~]# less /var/log/messages
...output omitted...
Feb 26 15:51:07 jegui NetworkManager[689]: <info>  [1551214267.8584] Loaded device
 plugin: NMTeamFactory (/usr/lib64/NetworkManager/1.14.0-14.el8/libnm-device-
plugin-team.so)
Feb 26 15:51:07 jegui NetworkManager[689]: <info>  [1551214267.8599] device (lo):
 carrier: link connected
Feb 26 15:51:07 jegui NetworkManager[689]: <info>  [1551214267.8600] manager:
 (lo): new Generic device (/org/freedesktop/NetworkManager/Devices/1)
Feb 26 15:51:07 jegui NetworkManager[689]: <info>  [1551214267.8623] manager:
 (ens3): new Ethernet device (/org/freedesktop/NetworkManager/Devices/2)
Feb 26 15:51:07 jegui NetworkManager[689]: <info>  [1551214267.8653] device
 (ens3): state change: unmanaged -> unavailable (reason 'managed', sys-iface-
state: 'external')
/device
```

REFERENCES

regex(7) and **grep**(1) man pages

GUIDED EXERCISE

MATCHING TEXT IN COMMAND OUTPUT WITH REGULAR EXPRESSIONS

In this lab, you will search for text in the system logs and the output of commands in order to find information more efficiently.

OUTCOMES

You should be able to efficiently search for text in log files and configuration files.

BEFORE YOU BEGIN

Log in to `workstation` as `student` using `student` as the password.

On `workstation`, run the **`lab console-regex start`** command. This command runs a start script that determines if the `servera` machine is reachable on the network. It also installs the *postfix* package.

```
[student@workstation ~]$ lab console-regex start
```

1. Use the **`ssh`** command to log in to `servera` as the `student` user. The systems are configured to use SSH keys for authentication, therefore a password is not required.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

2. Use the **`sudo -i`** command to switch to the `root` user. The password for the `student` user is `student`.

```
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

3. The *postfix* package was installed today by the `start` script. Use the **`grep`** command to find the GID and UID for the `postfix` and `postdrop` groups and users. To reduce the output of the **`grep`** command, display all logs from a specific **`Start Time`**.

   3.1. Use the **`date`** command to determine the current time.

```
[root@servera ~]# date
Fri Mar 22 08:23:56 CET 2019
```

   3.2. Use the **`grep`** command with the date, start time, and GID options to find the `postfix` and `postdrop` user's GID and UID. The lab set-up script ran a few minutes

before the current time. Take this into consideration when searching the **/var/log/secure** log file.

```
[root@servera ~]# grep '^Mar 22 08:2.*GID' /var/log/secure
Mar 22 08:20:04 servera groupadd[2514]: group added to /etc/
group: name=postdrop, GID=90
Mar 22 08:20:04 servera groupadd[2514]: new group: name=postdrop,
 GID=90
Mar 22 08:20:04 servera groupadd[2520]: group added to /etc/
group: name=postfix, GID=89
Mar 22 08:20:04 servera groupadd[2520]: new group: name=postfix,
 GID=89
Mar 22 08:20:04 servera useradd[2527]: new user: name=postfix, UID=89,
 GID=89, home=/var/spool/postfix, shell=/sbin/nologin
```

4. Modify your regular expression to locate the first two messages in the **/var/log/maillog** file. Notice that in this search you are not using the caret character (^) because you are not searching for the first character in a line.

```
[root@servera ~]# grep 'postfix' /var/log/maillog | head -n 2
Mar 22 08:21:02 servera postfix/postfix-script[3879]: starting the Postfix
 mail system
Mar 22 08:21:02 servera postfix/master[3881]: daemon started -- version
 3.3.1, configuration /etc/postfix
```

5. You are required to find the name of the **queue** directory for the `Postfix` server. Search the **/etc/postfix/main.cf** configuration file for all information about queues. Use the **-i** option to ignore case distinctions.

```
[root@servera ~]# grep -i 'queue' /etc/postfix/main.cf
# testing.  When soft_bounce is enabled, mail will remain queued that
# The queue_directory specifies the location of the Postfix queue.
queue_directory = /var/spool/postfix
# QUEUE AND PROCESS OWNERSHIP
# The mail_owner parameter specifies the owner of the Postfix queue
# is the Sendmail-compatible mail queue listing command.
# setgid_group: The group for mail submission and queue management
```

6. Confirm that `postfix` is writing messages to **/var/log/messages**. Use the **less** command then the slash character (/) to search the file. Press **n** to move to the next entry that matches the search. Use the **q** key to quit the **less** command.

```
[root@servera ~]# less /var/log/messages
...output omitted...
Mar 22 07:58:04 servera systemd[1]: Started Postfix Mail Transport Agent.
...output omitted...
Mar 22 08:12:26 servera systemd[1]: Stopping Postfix Mail Transport Agent...
Mar 22 08:12:26 servera systemd[1]: Stopped Postfix Mail Transport Agent.
...output omitted...
/Postfix
```

7. Use the **ps aux** command to confirm that the postfix server is currently running. Reduce the output of **ps aux** by combining it with the **grep** command.

```
[root@servera ~]# ps aux | grep postfix
root      3881  0.0  0.2 121664  5364 ?        Ss   08:21   0:00 /usr/
libexec/postfix/master -w
postfix   3882  0.0  0.4 147284  9088 ?        S    08:21   0:00 pickup -l -
t unix -u
postfix   3883  0.0  0.4 147336  9124 ?        S    08:21   0:00 qmgr -l -t
 unix -u
```

8. Confirm that the qmgr, cleanup, and pickup queues are correctly configured. Use the **grep** command with the **-e** option to match multiple entries in the same file. The configuration file is **/etc/postfix/master.cf**

```
[root@servera ~]# grep -e qmgr -e pickup -e cleanup /etc/postfix/master.cf
pickup    unix  n       -       n       60      1       pickup
cleanup   unix  n       -       n       -       0       cleanup
qmgr      unix  n       -       n       300     1       qmgr
#qmgr     unix  n       -       n       300     1       oqmgr
```

9. Log off from servera.

```
[root@servera ~]# exit
logout
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```

Finish

On workstation, run the **lab console-regex finish** script to complete this exercise.

```
[student@workstation ~]$ lab console-regex finish
```

This concludes the guided exercise.

LAB

IMPROVING COMMAND-LINE PRODUCTIVITY

PERFORMANCE CHECKLIST

In this lab, you will create a Bash script that can filter and get relevant information from different hosts.

OUTCOMES

You should be able to:

- Create a Bash script and redirect its output to a file.
- Use loops to simplify your code.
- Filter the relevant content using **grep** and regular expressions.

BEFORE YOU BEGIN

Log in to workstation as student using student as the password.

On workstation, run the **lab console-review start** command. This command runs a start script that determines if the workstation, servera and serverb machines are reachable on the network. The script will alert you if they are not available. It also installs the *vim-enhanced* and *util-linux* packages if needed, configures **sudo**, and prepares the content of **/var/log/secure** on servera and serverb.

```
[student@workstation ~]$ lab console-review start
```

1. Create the **/home/student/bin/bash-lab** script file on workstation.
2. Edit your newly created script file to comply with the following requested information from the servera and serverb hosts. The systems are configured to use SSH keys for authentication; a password is not required.

COMMAND OR FILE	CONTENT REQUESTED
hostname -f	Get all the output.
echo "#####"	Get all the output.
lscpu	Get only the lines that start with the string **CPU**.
echo "#####"	Get all the output.
/etc/selinux/config	Ignore empty lines. Ignore lines starting with #.

COMMAND OR FILE	CONTENT REQUESTED
echo "#####"	Get all the output.
/var/log/secure	Get all "Failed password" entries.
echo "#####"	Get all the output.

Save the required information to the new files **/home/student/output-servera** and **/home/student/output-serverb**.

> **NOTE**
> You can use **sudo** without requiring a password on the `servera` and `serverb` hosts. Remember to use a loop to simplify your script. You can also use multiple **grep** commands concatenated with the use of the pipe character (|).

3. Execute the **/home/student/bin/bash-lab** script, and review the output content on `workstation`.

Evaluation

On `workstation`, run the **lab console-review grade** command to confirm success of this exercise.

```
[student@workstation ~]$ lab console-review grade
```

Finish

On `workstation`, run the **lab console-review finish** script to complete this exercise.

```
[student@workstation ~]$ lab console-review finish
```

This concludes the lab.

SOLUTION

IMPROVING COMMAND-LINE PRODUCTIVITY

PERFORMANCE CHECKLIST

In this lab, you will create a Bash script that can filter and get relevant information from different hosts.

OUTCOMES

You should be able to:

- Create a Bash script and redirect its output to a file.
- Use loops to simplify your code.
- Filter the relevant content using **grep** and regular expressions.

BEFORE YOU BEGIN

Log in to workstation as student using student as the password.

On workstation, run the **lab console-review start** command. This command runs a start script that determines if the workstation, servera and serverb machines are reachable on the network. The script will alert you if they are not available. It also installs the *vim-enhanced* and *util-linux* packages if needed, configures **sudo**, and prepares the content of **/var/log/secure** on servera and serverb.

```
[student@workstation ~]$ lab console-review start
```

1. Create the **/home/student/bin/bash-lab** script file on workstation.

1.1. On workstation, create the **/home/student/bin/** folder if needed.

```
[student@workstation ~]$ mkdir -p /home/student/bin
```

1.2. Use **vim** to create and edit the **/home/student/bin/bash-lab** script file.

```
[student@workstation ~]$  vim ~/bin/bash-lab
```

1.3. Insert the following text and save the file.

```
#!/bin/bash
```

1.4. Make your script file executable.

```
[student@workstation ~]$ chmod a+x ~/bin/bash-lab
```

2. Edit your newly created script file to comply with the following requested information from the `servera` and `serverb` hosts. The systems are configured to use SSH keys for authentication; a password is not required.

COMMAND OR FILE	CONTENT REQUESTED
`hostname -f`	Get all the output.
`echo "#####"`	Get all the output.
`lscpu`	Get only the lines that start with the string **`CPU`**.
`echo "#####"`	Get all the output.
`/etc/selinux/config`	Ignore empty lines. Ignore lines starting with #.
`echo "#####"`	Get all the output.
`/var/log/secure`	Get all "Failed password" entries.

COMMAND OR FILE	CONTENT REQUESTED
echo "#####"	Get all the output.

Save the required information to the new files **/home/student/output-servera** and **/home/student/output-serverb**.

NOTE
You can use **sudo** without requiring a password on the `servera` and `serverb` hosts. Remember to use a loop to simplify your script. You can also use multiple **grep** commands concatenated with the use of the pipe character (|).

2.1. Use **vim** to open and edit the **/home/student/bin/bash-lab** script file.

```
[student@workstation ~]$ vim ~/bin/bash-lab
```

2.2. Append the following lines in bold to the **/home/student/bin/bash-lab** script file.

NOTE
The following is an example of how you can achieve the requested script. In Bash scripting, you can take different approaches and obtain the same result.

```
#!/bin/bash
#
USR='student'
OUT='/home/student/output'
#
for SRV in servera serverb
  do
ssh ${USR}@${SRV} "hostname -f" > ${OUT}-${SRV}
echo "#####" >> ${OUT}-${SRV}
ssh ${USR}@${SRV} "lscpu | grep '^CPU'" >> ${OUT}-${SRV}
echo "#####" >> ${OUT}-${SRV}
ssh ${USR}@${SRV} "grep -v '^$' /etc/selinux/config|grep -v '^#'" >>
 ${OUT}-${SRV}
echo "#####" >> ${OUT}-${SRV}
ssh ${USR}@${SRV} "sudo grep 'Failed password' /var/log/secure" >>
 ${OUT}-${SRV}
echo "#####" >> ${OUT}-${SRV}
done
```

3. Execute the **/home/student/bin/bash-lab** script, and review the output content on workstation.

   3.1. On workstation, execute the **/home/student/bin/bash-lab** script.

```
[student@workstation ~]$ bash-lab
```

   3.2. Review the content of **/home/student/output-servera** and **/home/student/output-serverb**.

```
[student@workstation ~]$ cat /home/student/output-servera
servera.lab.example.com
#####
CPU op-mode(s):      32-bit, 64-bit
CPU(s):              2
CPU family:          21
CPU MHz:             2294.670
#####
SELINUX=enforcing
SELINUXTYPE=targeted
#####
Mar 21 22:30:28 servera sshd[3939]: Failed password for invalid user
 operator1 from 172.25.250.9 port 58382 ssh2
Mar 21 22:30:31 servera sshd[3951]: Failed password for invalid user
 sysadmin1 from 172.25.250.9 port 58384 ssh2
Mar 21 22:30:34 servera sshd[3953]: Failed password for invalid user
 manager1 from 172.25.250.9 port 58386 ssh2
#####
```

```
[student@workstation ~]$ cat /home/student/output-serverb
serverb.lab.example.com
#####
CPU op-mode(s):      32-bit, 64-bit
CPU(s):              2
CPU family:          6
CPU MHz:             2294.664
#####
SELINUX=enforcing
SELINUXTYPE=targeted
#####
Mar 21 22:30:37 serverb sshd[3883]: Failed password for invalid user
 operator1 from 172.25.250.9 port 39008 ssh2
Mar 21 22:30:39 serverb sshd[3891]: Failed password for invalid user
 sysadmin1 from 172.25.250.9 port 39010 ssh2
Mar 21 22:30:43 serverb sshd[3893]: Failed password for invalid user
 manager1 from 172.25.250.9 port 39012 ssh2
#####
```

Evaluation

On workstation, run the **lab console-review grade** command to confirm success of this exercise.

```
[student@workstation ~]$ lab console-review grade
```

Finish

On `workstation`, run the **`lab console-review finish`** script to complete this exercise.

```
[student@workstation ~]$ lab console-review finish
```

This concludes the lab.

SUMMARY

In this chapter, you learned:

- How to create and execute simple Bash scripts.
- How to use loops to iterate through a list of items from the command-line and in a shell script.
- How to search for text in log files and configuration files using regular expressions and **`grep`**.

CHAPTER 2

SCHEDULING FUTURE TASKS

GOAL

Schedule tasks to automatically execute in the future.

OBJECTIVES

- Set up a command that runs once at some point in the future.
- Schedule commands to run on a repeating schedule using a user's crontab file.
- Schedule commands to run on a repeating schedule using the system crontab file and directories.
- Enable and disable systemd timers, and configure a timer that manages temporary files.

SECTIONS

- Scheduling a Deferred User Job (and Guided Exercise)
- Scheduling Recurring User Jobs (and Guided Exercise)
- Scheduling Recurring System Jobs (and Guided Exercise)
- Managing Temporary Files (and Guided Exercise)

LAB

Scheduling Future Tasks

SCHEDULING A DEFERRED USER JOB

OBJECTIVES

After completing this section, you should be able to set up a command that runs once at some point in the future.

DESCRIBING DEFERRED USER TASKS

Sometimes you might need to run a command, or set of commands, at a set point in the future. Examples include people who want to schedule an email to their boss, or a system administrator working on a firewall configuration who puts a "safety" job in place to reset the firewall settings in ten minutes' time, unless they deactivate the job beforehand.

These scheduled commands are often called *tasks* or *jobs*, and the term *deferred* indicates that these tasks or jobs are going to run in the future.

One of the solutions available to Red Hat Enterprise Linux users for scheduling deferred tasks is *at*. The *at* package provides the (`atd`) system daemon along with a set of command-line tools to interact with the daemon (**`at`**, **`atq`**, and more). In a default Red Hat Enterprise Linux installation, the `atd` daemon is installed and enabled automatically.

Users (including `root`) can queue up jobs for the `atd` daemon using the **`at`** command. The `atd` daemon provides 26 queues, **`a`** to **`z`**, with jobs in alphabetically later queues getting lower system priority (higher *nice* values, discussed in a later chapter).

Scheduling Deferred User Tasks

Use the **`at`** ***`TIMESPEC`*** command to schedule a new job. The **`at`** command then reads the commands to execute from the `stdin` channel. While manually entering commands, you can finish your input by pressing **`Ctrl`**+**`D`**. For more complex commands that are prone to typographical errors, it is often easier to use input redirection from a script file, for example, **`at now +5min <`** ***`myscript`***, rather than typing all the commands manually in a terminal window.

The ***`TIMESPEC`*** argument with the **`at`** command accepts many powerful combinations, allowing users to describe exactly when a job should run. Typically, they start with a time, for example, **`02:00pm`**, **`15:59`**, or even **`teatime`**, followed by an optional date or number of days in the future. The following lists some examples of combinations that can be used.

- **`now +5min`**
- **`teatime tomorrow`** (teatime is **`16:00`**)
- **`noon +4 days`**
- **`5pm august 3 2021`**

For a complete list of valid time specifications, refer to the **`timespec`** definition as listed in the references.

INSPECTING AND MANAGING DEFERRED USER JOBS

To get an overview of the pending jobs for the current user, use the command **atq** or the **at -l** commands.

```
[user@host ~]$ atq
❶28  ❷Mon Feb  2 05:13:00 2015 ❸a ❹user
29  Mon Feb  3 16:00:00 2014 h user
27  Tue Feb  4 12:00:00 2014 a user
```

In the preceding output, every line represents a different job scheduled to run in the future.

❶ The unique job number for this job.
❷ The execution date and time for the scheduled job.
❸ Indicates that the job is scheduled with the default queue `a`. Different jobs may be scheduled with different queues.
❹ The owner of the job (and the user that the job will run as).

IMPORTANT
Unprivileged users can only see and control their own jobs. The `root` user can see and manage all jobs.

To inspect the actual commands that will run when a job is executed, use the **at -c *JOBNUMBER*** command. This command shows the *environment* for the job being set up to reflect the environment of the user who created the job at the time it was created, followed by the actual commands to be run.

Removing Jobs

The **atrm *JOBNUMBER*** command removes a scheduled job. Remove the scheduled job when it is no longer needed, for example, when a remote firewall configuration succeeded, and does not need to be reset.

REFERENCES
at(1) and **atd**(8) man pages

/usr/share/doc/at/timespec

GUIDED EXERCISE

SCHEDULING A DEFERRED USER JOB

In this exercise, you will use the **at** command to schedule several commands to run at specified times in the future.

OUTCOMES

You should be able to:

- Schedule a job to run at a specified time in the future.
- Inspect the commands that a scheduled job runs.
- Delete the scheduled jobs.

BEFORE YOU BEGIN

Log in to `workstation` as `student` using `student` as the password.

On `workstation`, run **`lab scheduling-at start`** to start the exercise. This script ensures that the environment is clean and set up correctly.

```
[student@workstation ~]$ lab scheduling-at start
```

1. From `workstation`, open an SSH session to `servera` as `student`.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

2. Schedule a job to run in three minutes from now using the **at** command. The job must save the output of the **date** command to **/home/student/myjob.txt**.

   2.1. Use the **echo** command to pass the string **date >> /home/student/myjob.txt** as input to the **at** command so that the job runs in three minutes from now.

```
[student@servera ~]$ echo "date >> /home/student/myjob.txt" | at now
 +3min
warning: commands will be executed using /bin/sh
job 1 at Thu Mar 21 12:30:00 2019
```

   2.2. Use the **atq** command to list the scheduled jobs.

```
[student@servera ~]$ atq
1 Thu Mar 21 12:30:00 2019 a student
```

2.3. Use the **watch atq** command to command to monitor the queue of the deferred jobs in real time. The job is removed from the queue after its execution.

```
[student@servera ~]$ watch atq
Every 2.0s: atq         servera.lab.example.com: Thu Mar 21 12:30:00
 2019

1 Thu Mar 21 12:30:00 2019 a student
```

The preceding **watch** command updates the output of **atq** every two seconds, by default. After the deferred job is removed from the queue, press **Ctrl**+**c** to exit **watch** and return to the shell prompt.

2.4. Use the **cat** command to verify that the contents of **/home/student/myjob.txt** match the output of the **date** command.

```
[student@servera ~]$ cat myjob.txt
Thu Mar 21 12:30:00 IST 2019
```

The preceding output matches with the output of the **date** command, confirming that the scheduled job executed successfully.

3. Use the **at** command to interactively schedule a job with the queue **g** that runs at **teatime** (16:00). The job should execute a command that prints the message **It's teatime** to **/home/student/tea.txt**. The new messages should be appended to the file **/home/student/tea.txt**.

```
[student@servera ~]$ at -q g teatime
warning: commands will be executed using /bin/sh
at> echo "It's teatime" >> /home/student/tea.txt
at> Ctrl+d
job 2 at Thu Mar 21 16:00:00 2019
```

4. Use the **at** command to interactively schedule another job with the queue **b** that runs at **16:05**. The job should execute a command that prints the message **The cookies are good** to **/home/student/cookies.txt**. The new messages should be appended to the file **/home/student/cookies.txt**.

```
[student@servera ~]$ at -q b 16:05
warning: commands will be executed using /bin/sh
at> echo "The cookies are good" >> /home/student/cookies.txt
at> Ctrl+d
job 3 at Thu Mar 21 16:05:00 2019
```

5. Inspect the commands in the pending jobs.

5.1. Use the **atq** command to view the job numbers of the pending jobs.

```
[student@servera ~]$ atq
2 Thu Mar 21 16:00:00 2019 g student
3 Thu Mar 21 16:05:00 2019 b student
```

Note the job numbers in the preceding output. These job numbers may vary on your system.

5.2. Use the **at** command to view the commands in the pending job number 2.

```
[student@servera ~]$ at -c 2
...output omitted...
echo "It's teatime" >> /home/student/tea.txt
marcinDELIMITER28d54caa
```

Notice that the preceding scheduled job executes an **echo** command that appends the message **It's teatime** to **/home/student/tea.txt**.

5.3. Use the **at** command to view the commands in the pending job number 3.

```
[student@servera ~]$ at -c 3
...output omitted...
echo "The cookies are good" >> /home/student/cookies.txt
marcinDELIMITER1d2b47e9
```

Notice that the preceding scheduled job executes an **echo** command that appends the message **The cookies are good** to **/home/student/cookies.txt**.

6. Use the **atq** command to view the job number of a job that runs at **teatime** (16:00) and remove it using the **atrm** command.

```
[student@servera ~]$ atq
2 Thu Mar 21 16:00:00 2019 g student
3 Thu Mar 21 16:05:00 2019 b student
[student@servera ~]$ atrm 2
```

7. Verify that the job scheduled to run at **teatime** (16:00) no longer exists.

   7.1. Use the **atq** command to view the list of pending jobs and confirm that the job scheduled to run at **teatime** (16:00) no longer exists.

```
[student@servera ~]$ atq
3 Thu Mar 21 16:05:00 2019 b student
```

   7.2. Log off from servera.

```
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```

Finish

On workstation, run **lab scheduling-at finish** to complete this exercise. This script deletes the files created throughout the exercise and ensures that the environment is clean.

```
[student@workstation ~]$ lab scheduling-at finish
```

This concludes the guided exercise.

SCHEDULING RECURRING USER JOBS

OBJECTIVES

After completing this section, you should be able to schedule commands to run on a repeating schedule using a user's crontab file.

DESCRIBING RECURRING USER JOBS

Jobs scheduled to run repeatedly are called *recurring jobs*. Red Hat Enterprise Linux systems ship with the `crond` daemon, provided by the *cronie* package, enabled and started by default specifically for recurring jobs. The `crond` daemon reads multiple configuration files: one per user (edited with the **`crontab`** command), and a set of system-wide files. These configuration files give users and administrators fine-grained control over when their recurring jobs should be executed.

If a scheduled command produces any output or error that is not redirected, the `crond` daemon attempts to email that output or error to the user who owns that job (unless overridden) using the mail server configured on the system. Depending on the environment, this may need additional configuration. The output or error of the scheduled command can be redirected to different files.

SCHEDULING RECURRING USER JOBS

Normal users can use the **`crontab`** command to manage their jobs. This command can be called in four different ways:

Crontab Examples

COMMAND	INTENDED USE
`crontab -l`	List the jobs for the current user.
`crontab -r`	Remove all jobs for the current user.
`crontab -e`	Edit jobs for the current user.
`crontab` ***`filename`***	Remove all jobs, and replace with the jobs read from *filename*. If no file is specified, **`stdin`** is used.

NOTE
The superuser can use the **`-u`** option with the **`crontab`** command to manage jobs for another user. You should not use the **`crontab`** command to manage system jobs; instead, use the methods described in the next section.

DESCRIBING USER JOB FORMAT

The **`crontab -e`** command invokes Vim by default, unless the `EDITOR` environment variable has been set to something different. Enter one job per line. Other valid entries include: empty lines,

typically for ease of reading; comments, identified by lines starting with the number sign (#); and environment variables using the format **`NAME=value`**, which affects all lines below the line where they are declared. Common variable settings include the `SHELL` variable, which declares which shell to use to interpret the remaining lines of the crontab file; and the `MAILTO` variable, which determines who should receive any emailed output.

IMPORTANT
Sending email may require additional configuration of the local mail server or SMTP relay on a system.

Fields in the **`crontab`** file appear in the following order:

- Minutes
- Hours
- Day of month
- Month
- Day of week
- Command

IMPORTANT
When the **`Day of month`** and **`Day of week`** fields are both other than **`*`**, the command is executed when either of these two fields are satisfied. For example, to run a command on the 15th of every month, and every Friday at **`12:15`**, use the following job format:

```
15 12 15 * Fri command
```

The first five fields all use the same syntax rules:

- **`*`** for "Do not Care"/always.
- A number to specify a number of minutes or hours, a date, or a weekday. For weekdays, **`0`** equals Sunday, **`1`** equals Monday, **`2`** equals Tuesday, and so on. **`7`** also equals Sunday.
- **`x-y`** for a range, **`x`** to **`y`** inclusive.
- **`x,y`** for lists. Lists can include ranges as well, for example, **`5,10-13,17`** in the **`Minutes`** column to indicate that a job should run at 5, 10, 11, 12, 13, and 17 minutes past the hour.
- **`*/x`** to indicate an interval of **`x`**, for example, **`*/7`** in the **`Minutes`** column runs a job every seven minutes.

Additionally, 3-letter English abbreviations can be used for both months and weekdays, for example, Jan, Feb, and Mon, Tue.

The last field contains the command to execute using the default shell. The **`SHELL`** environment variable can used to change the shell for the scheduled command. If the command contains an unescaped percentage sign (%), then that percentage sign is treated as a newline character, and everything after the percentage sign is passed to the command on **`stdin`**.

Example Recurring User Jobs

This section describes some examples of recurring jobs.

- The following job executes the command **/usr/local/bin/yearly_backup** at exactly 9 a.m. on February 2nd, every year.

```
0 9 2 2 * /usr/local/bin/yearly_backup
```

- The following job sends an email containing the word **Chime** to the owner of this job, every five minutes between 9 a.m. and 5 p.m., on every Friday in July.

```
*/5 9-16 * Jul 5 echo "Chime"
```

  The preceding **9-16** range of hours means that the job timer starts at the ninth hour (09:00) and continues until the end of the sixteenth hour (16:59). The job starts executing at **09:00** with the last execution at **16:55** because five minutes from **16:55** is **17:00** which is beyond the given scope of hours.

- The following job runs the command **/usr/local/bin/daily_report** every weekday at two minutes before midnight.

```
58 23 * * 1-5 /usr/local/bin/daily_report
```

- The following job executes the **mutt** command to send the mail message **Checking in** to the recipient **boss@example.com** on every workday (Monday to Friday), at 9 a.m.

```
0 9 * * 1-5 mutt -s "Checking in" boss@example.com % Hi there boss, just
 checking in.
```

REFERENCES

crond(8), **crontab**(1), and **crontab**(5) man pages

GUIDED EXERCISE

SCHEDULING RECURRING USER JOBS

In this exercise, you will schedule commands to run on a repeating schedule as a non-privileged user, using the **`crontab`** command.

OUTCOMES

You should be able to:

- Schedule recurring jobs to run as a non-privileged user.
- Inspect the commands that a scheduled recurring job runs.
- Remove scheduled recurring jobs.

BEFORE YOU BEGIN

Log in to `workstation` as `student` using `student` as the password.

On `workstation`, run **`lab scheduling-cron start`** to start the exercise. This script ensures that the environment is clean and set up correctly.

```
[student@workstation ~]$ lab scheduling-cron start
```

1. From `workstation`, open an SSH session to `servera` as `student`.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

2. Schedule a recurring job as `student` that appends the current date and time to **`/home/student/my_first_cron_job.txt`** every two minutes between 8 a.m. and 9 p.m. The job must only run from Monday to Friday, not on Saturday or Sunday.

> **IMPORTANT**
> If you are working on this lab outside of the day and time mentioned in the preceding instruction, you should adjust your system time and/or date accordingly so that the job runs while you are working.

2.1. Use the **`crontab -e`** command to open the crontab using the default text editor.

```
[student@servera ~]$ crontab -e
```

2.2. Insert the following line.

```
*/2 08-20 * * Mon-Fri /usr/bin/date >> /home/student/
my_first_cron_job.txt
```

2.3. While in the text editor, press **Esc** and type **:wq** to save the changes and exit the editor. When the editor exits, you should see the following output:

```
...output omitted...
crontab: installing new crontab
[student@servera ~]$
```

The preceding output confirms that the job was scheduled successfully.

3. Use the **crontab -l** command to list the scheduled recurring jobs. Inspect the command you scheduled to run as a recurring job in the preceding step.

```
[student@servera ~]$ crontab -l
*/2 08-20 * * * /usr/bin/date >> /home/student/my_first_cron_job.txt
```

Notice that the preceding scheduled job runs the **/usr/bin/date** command and appends its output to **/home/student/my_first_cron_job.txt**.

4. Use the **while** command so that your shell prompt sleeps until the **/home/student/my_first_cron_job.txt** file is created as a result of the successful execution of the recurring job you scheduled. Wait for your shell prompt to return.

```
[student@servera ~]$ while ! test -f my_first_cron_job.txt; do sleep 1s;
 done
```

The preceding **while** command uses **! test -f** to continue running a loop of **sleep 1s** commands until the **my_first_cron_job.txt** file is created in the **/home/student** directory.

5. Use the **cat** command to verify that the contents of **/home/student/my_first_cron_job.txt** match the output of the **date** command.

```
[student@servera ~]$ cat my_first_cron_job.txt
Fri Mar 22 13:56:01 IST 2019
```

The preceding output may vary on your system.

6. Remove all the recurring jobs scheduled to run as `student`.

   6.1. Use the **`crontab -r`** command to remove all the scheduled recurring jobs for `student`.

   ```
   [student@servera ~]$ crontab -r
   ```

   6.2. Use the **`crontab -l`** command to verify that no recurring jobs exist for `student`.

   ```
   [student@servera ~]$ crontab -l
   no crontab for student
   ```

   6.3. Log off from `servera`.

   ```
   [student@servera ~]$ exit
   logout
   Connection to servera closed.
   [student@workstation ~]$
   ```

Finish

On `workstation`, run **`lab scheduling-cron finish`** to complete this exercise. This script deletes the files created throughout the exercise and ensures that the environment is clean.

```
[student@workstation ~]$ lab scheduling-cron finish
```

This concludes the guided exercise.

SCHEDULING RECURRING SYSTEM JOBS

OBJECTIVES

After completing this section, you should be able to schedule commands to run on a repeating schedule using the system crontab file and directories.

DESCRIBING RECURRING SYSTEM JOBS

System administrators often need to run recurring jobs. Best practice is to run these jobs from system accounts rather than from user accounts. That is, do not schedule to run these jobs using the **crontab** command, but instead use system-wide crontab files. Job entries in the system-wide crontab files are similar to those of the users' crontab entries, excepting only that the system-wide crontab files have an extra field before the command field; the user under whose authority the command should run.

The **/etc/crontab** file has a useful syntax diagram in the included comments.

```
 # For details see man 4 crontabs

# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue ...
# |  |  |  |  |
# *  *  *  *  * user-name  command to be executed
```

Recurring system jobs are defined in two locations: the **/etc/crontab** file, and files within the **/etc/cron.d/** directory. You should always create your custom crontab files under the **/etc/cron.d** directory to schedule recurring system jobs. Place the custom crontab file in **/etc/cron.d** to protect it from being overwritten if any package update occurs to the provider of **/etc/crontab**, which may overwrite the existing contents in **/etc/crontab**. Packages that require recurring system jobs place their crontab files in **/etc/cron.d/** containing the job entries. Administrators also use this location to group related jobs into a single file.

The crontab system also includes repositories for scripts that need to run every hour, day, week, and month. These repositories are directories called **/etc/cron.hourly/**, **/etc/cron.daily/**, **/etc/cron.weekly/**, and **/etc/cron.monthly/**. Again, these directories contain executable shell scripts, not crontab files.

IMPORTANT

Remember to make any script you place in these directories executable. If a script is not executable, it will not run. To make a script executable, use the **chmod +x *script_name***.

A command called **run-parts** called from the **/etc/cron.d/0hourly** file runs the **/etc/cron.hourly/*** scripts. The **run-parts** command also runs the daily, weekly, and monthly jobs, but it is called from a different configuration file called **/etc/anacrontab**.

> **NOTE**
> In the past, a separate service called anacron used to handle the **/etc/anacrontab** file, but in Red Hat Enterprise Linux 7 and later, the regular crond service parses this file.

The purpose of **/etc/anacrontab** is to make sure that important jobs always run, and not skipped accidentally because the system was turned off or hibernating when the job should have been executed. For example, if a system job that runs daily was not executed last time it was due because the system was rebooting, the job is executed when the system becomes ready. However, there may be a delay of several minutes in starting the job depending on the value of the **Delay in minutes** parameter specified for the job in **/etc/anacrontab**.

There are different files in **/var/spool/anacron/** for each of the daily, weekly, and monthly jobs to determine if a particular job has run. When crond starts a job from **/etc/anacrontab**, it updates the time stamps of those files. The same time stamp is used to determine when a job was last run. The syntax of **/etc/anacrontab** is different from the regular **crontab** configuration files. It contains exactly four fields per line, as follows.

- **Period in days**

  The interval in days for the job that runs on a repeating schedule. This field accepts an integer or a macro as its value. For example, the macro **@daily** is equivalent to the integer **1**, which means that the job is executed on a daily basis. Similarly, the macro **@weekly** is equivalent to the integer **7**, which means that the job is executed on a weekly basis.

- **Delay in minutes**

  The amount of time the crond daemon should wait before starting this job.

- **Job identifier**

  The unique name the job is identified as in the log messages.

- **Command**

  The command to be executed.

The **/etc/anacrontab** file also contains environment variable declarations using the syntax **NAME=value**. Of special interest is the variable START_HOURS_RANGE, which specifies the time interval for the jobs to run. Jobs are not started outside of this range. If on a particular day, a job does not run within this time interval, the job has to wait until the next day for execution.

INTRODUCING SYSTEMD TIMER

With the advent of systemd in Red Hat Enterprise Linux 7, a new scheduling function is now available: systemd *timer units*. A systemd timer unit activates another unit of a different type (such as a service) whose unit name matches the timer unit name. The timer unit allows timer-based activation of other units. For easier debugging, systemd logs timer events in system journals.

Sample Timer Unit

The *sysstat* package provides a `systemd` timer unit called `sysstat-collect.timer` to collect system statistics every 10 minutes. The following output shows the configuration lines of **`/usr/lib/systemd/system/sysstat-collect.timer`**.

```
...output omitted...
[Unit]
Description=Run system activity accounting tool every 10 minutes

[Timer]
OnCalendar=*:00/10

[Install]
WantedBy=sysstat.service
```

The parameter **`OnCalendar=*:00/10`** signifies that this timer unit activates the corresponding unit (`sysstat-collect.service`) every 10 minutes. However, you can specify more complex time intervals. For example, a value of **`2019-03-* 12:35,37,39:16`** against the **`OnCalendar`** parameter causes the timer unit to activate the corresponding service unit at **`12:35:16`**, **`12:37:16`**, and **`12:39:16`** every day throughout the entire month of March, 2019. You can also specify relative timers using parameters such as **`OnUnitActiveSec`**. For example, the **`OnUnitActiveSec=15min`** option causes the timer unit to trigger the corresponding unit 15 minutes after the last time the timer unit activated its corresponding unit.

IMPORTANT

Do not modify any unit configuration file under the **`/usr/lib/systemd/system`** directory because any update to the provider package of the configuration file may override the configuration changes you made in that file. So, make a copy of the unit configuration file you intend to change under the **`/etc/systemd/system`** directory and then modify the copy so that the configuration changes you make with respect to a unit does not get overridden by any update to the provider package. If two files exist with the same name under the **`/usr/lib/systemd/system`** and **`/etc/systemd/system`** directories, `systemd` parses the file under the **`/etc/systemd/system`** directory.

After you change the timer unit configuration file, use the **`systemctl daemon-reload`** command to ensure that `systemd` is aware of the changes. This command reloads the `systemd` manager configuration.

```
[root@host ~]# systemctl daemon-reload
```

After you reload the `systemd` manager configuration, use the following **`systemctl`** command to activate the timer unit.

```
[root@host ~]# systemctl enable --now <unitname>.timer
```

REFERENCES

`crontab`(5), **`anacron`**(8), **`anacrontab`**(5), `systemd.time`(7), `systemd.timer`(5), and `crond`(8) man pages

GUIDED EXERCISE

SCHEDULING RECURRING SYSTEM JOBS

In this exercise, you will schedule commands to run on various schedules by adding configuration files to the system crontab directories.

OUTCOMES

You should be able to:

- Schedule a recurring system job to count the number of active users.
- Update the `systemd` timer unit that gathers system activity data.

BEFORE YOU BEGIN

Log in to `workstation` as `student` using `student` as the password.

On `workstation`, run **`lab scheduling-system start`** to start the exercise. This script ensures that the environment is clean and set up correctly.

```
[student@workstation ~]$ lab scheduling-system start
```

1. From `workstation`, open an SSH session to `servera` as `student`.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

2. Use the **`sudo -i`** command to switch to the `root` user's account.

```
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

3. Schedule a recurring system job that generates a log message indicating the number of currently active users in the system. The job must run daily. You can use the **`w -h | wc -l`** command to retrieve the number of currently active users in the system. Also, use the **`logger`** command to generate the log message.

   3.1. Create a script file called **`/etc/cron.daily/usercount`** with the following content. You can use the **`vi /etc/cron.daily/usercount`** command to create the script file.

```
#!/bin/bash
USERCOUNT=$(w -h | wc -l)
logger "There are currently ${USERCOUNT} active users"
```

3.2. Use the **chmod** command to enable the execute (**x**) permission on **/etc/cron.daily/usercount**.

```
[root@servera ~]# chmod +x /etc/cron.daily/usercount
```

4. The *sysstat* package provides the `systemd` units called `sysstat-collect.timer` and `sysstat-collect.service`. The timer unit triggers the service unit every 10 minutes to collect system activity data using the shell script called **/usr/lib64/sa/sa1**. Make sure that the *sysstat* package is installed and change the timer unit configuration file to collect the system activity data every two minutes.

4.1. Use the **yum** command to install the *sysstat* package.

```
[root@servera ~]# yum install sysstat
...output omitted...
Is this ok [y/N]: y
...output omitted...
Installed:
  sysstat-11.7.3-2.el8.x86_64             lm_sensors-
libs-3.4.0-17.20180522git70f7e08.el8.x86_64

Complete!
```

4.2. Copy **/usr/lib/systemd/system/sysstat-collect.timer** to **/etc/systemd/system/sysstat-collect.timer**.

```
[root@servera ~]# cp /usr/lib/systemd/system/sysstat-collect.timer \
/etc/systemd/system/sysstat-collect.timer
```

IMPORTANT

You should not edit files under the **/usr/lib/systemd** directory. With `systemd`, you can copy the unit file to the **/etc/systemd/system** directory and edit that copy. The `systemd` process parses your customized copy instead of the file under the **/usr/lib/systemd** directory.

4.3. Edit **/etc/systemd/system/sysstat-collect.timer** so that the timer unit runs every two minutes. Also, replace any occurrence of the string **10 minutes** with **2 minutes** throughout the unit configuration file including the ones in the commented lines. You may use the **vi /etc/systemd/system/sysstat-collect.timer** command to edit the configuration file.

```
...
#        Activates activity collector every 2 minutes

[Unit]
```

```
Description=Run system activity accounting tool every 2 minutes

[Timer]
OnCalendar=*:00/02

[Install]
WantedBy=sysstat.service
```

The preceding changes cause the `sysstat-collect.timer` unit to trigger `sysstat-collect.service` unit every two minutes, which runs **`/usr/lib64/sa/sa1 1 1`**. Running **`/usr/lib64/sa/sa1 1 1`** collects the system activity data in a binary file under the **`/var/log/sa`** directory.

4.4. Use the **`systemctl daemon-reload`** command to make sure that `systemd` is aware of the changes.

```
[root@servera ~]# systemctl daemon-reload
```

4.5. Use the **`systemctl`** command to activate the `sysstat-collect.timer` timer unit.

```
[root@servera ~]# systemctl enable --now sysstat-collect.timer
```

4.6. Use the **`while`** command to wait until the binary file gets created under the **`/var/log/sa`** directory. Wait for your shell prompt to return.

```
[root@servera ~]# while [ $(ls /var/log/sa | wc -l) -eq 0 ]; do sleep
 1s; done
```

In the **`while`** command above, the **`ls /var/log/sa | wc -l`** returns a **`0`** if the file does not exist and a **`1`** if it does exist. The **`while`** determines if this equals **`0`** and

if so, enters the loop, which pauses for one second. When the file exists, the **while** loop exits.

4.7. Use the **ls -l** command to verify that the binary file under the **/var/log/sa** directory got modified within last two minutes.

```
[root@servera ~]# ls -l /var/log/sa
total 8
-rw-r--r--. 1 root root 5156 Mar 25 12:34 sa25
[root@servera ~]# date
Mon Mar 25 12:35:32 +07 2019
```

The output of the preceding commands may vary on your system.

4.8. Exit the `root` user's shell.

```
[root@servera ~]# exit
logout
[student@servera ~]$
```

4.9. Log off from `servera`.

```
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```

Finish

On `workstation`, run **lab scheduling-system finish** to complete this exercise. This script deletes the files created throughout the exercise and ensures that the environment is clean.

```
[student@workstation ~]$ lab scheduling-system finish
```

This concludes the guided exercise.

MANAGING TEMPORARY FILES

OBJECTIVES

After completing this section, you should be able to enable and disable `systemd` timers, and configure a timer that manages temporary files.

MANAGING TEMPORARY FILES

A modern system requires a large number of temporary files and directories. Some applications (and users) use the **`/tmp`** directory to hold temporary data, while others use a more task-specific location such as daemon and user-specific *volatile* directories under **`/run`**. In this context, volatile means that the file system storing these files only exists in memory. When the system reboots or loses power, all the contents of volatile storage will be gone.

To keep a system running cleanly, it is necessary for these directories and files to be created when they do not exist, because daemons and scripts might rely on them being there, and for old files to be purged so that they do not fill up disk space or provide faulty information.

Red Hat Enterprise Linux 7 and later include a new tool called **`systemd-tmpfiles`**, which provides a structured and configurable method to manage temporary directories and files.

When `systemd` starts a system, one of the first service units launched is `systemd-tmpfiles-setup`. This service runs the command **`systemd-tmpfiles --create --remove`**. This command reads configuration files from **`/usr/lib/tmpfiles.d/*.conf`**, **`/run/tmpfiles.d/*.conf`**, and **`/etc/tmpfiles.d/*.conf`**. Any files and directories marked for deletion in those configuration files is removed, and any files and directories marked for creation (or permission fixes) will be created with the correct permissions if necessary.

Cleaning Temporary Files with a Systemd Timer

To ensure that long-running systems do not fill up their disks with stale data, a `systemd` timer unit called `systemd-tmpfiles-clean.timer` triggers `systemd-tmpfiles-clean.service` on a regular interval, which executes the **`systemd-tmpfiles --clean`** command.

The **`systemd`** timer unit configuration files have a **`[Timer]`** section that indicates how often the service with the same name should be started.

Use the following **`systemctl`** command to view the contents of the **`systemd-tmpfiles-clean.timer`** unit configuration file.

```
[user@host ~]$ systemctl cat systemd-tmpfiles-clean.timer
# /usr/lib/systemd/system/systemd-tmpfiles-clean.timer
#  SPDX-License-Identifier: LGPL-2.1+
#
#  This file is part of systemd.
#
#  systemd is free software; you can redistribute it and/or modify it
#  under the terms of the GNU Lesser General Public License as published
#  by
#  the Free Software Foundation; either version 2.1 of the License, or
```

```
#  (at your option) any later version.

[Unit]
Description=Daily Cleanup of Temporary Directories
Documentation=man:tmpfiles.d(5) man:systemd-tmpfiles(8)

[Timer]
OnBootSec=15min
```

In the preceding configuration the parameter **OnBootSec=15min** indicates that the service unit called `systemd-tmpfiles-clean.service` gets triggered 15 minutes after the system has booted up. The parameter **OnUnitActiveSec=1d** indicates that any further trigger to the `systemd-tmpfiles-clean.service` service unit happens 24 hours after the service unit was activated last.

Based on your requirement, you can change the parameters in the **systemd-tmpfiles-clean.timer** timer unit configuration file. For example, the value **30min** for the parameter **OnUnitActiveSec** triggers the `systemd-tmpfiles-clean.service` service unit 30 minutes after the service unit was last activated. As a result, `systemd-tmpfiles-clean.service` gets triggered every 30 minutes after bringing the changes into effect.

After changing the timer unit configuration file, use the **systemctl daemon-reload** command to ensure that `systemd` is aware of the change. This command reloads the `systemd` manager configuration.

```
[root@host ~]# systemctl daemon-reload
```

After you reload the `systemd` manager configuration, use the following **systemctl** command to activate the `systemd-tmpfiles-clean.timer` unit.

```
[root@host ~]# systemctl enable --now systemd-tmpfiles-clean.timer
```

Cleaning Temporary Files Manually

The command **systemd-tmpfiles --clean** parses the same configuration files as the **systemd-tmpfiles --create** command, but instead of creating files and directories, it will purge all files which have not been accessed, changed, or modified more recently than the maximum age defined in the configuration file.

The format of the configuration files for **systemd-tmpfiles** is detailed in the **tmpfiles.d**(5) manual page. The basic syntax consists of seven columns: Type, Path, Mode, UID, GID, Age, and Argument. *Type* refers to the action that **systemd-tmpfiles** should take; for example, **d** to create a directory if it does not yet exist, or **Z** to recursively restore SELinux contexts and file permissions and ownership.

The following are some examples with explanations.

```
d /run/systemd/seats 0755 root root -
```

When creating files and directories, create the **/run/systemd/seats** directory if it does not yet exist, owned by the user `root` and the group `root`, with permissions set to **rwxr-xr-x**. This directory will not be automatically purged.

```
D /home/student 0700 student student 1d
```

Create the **/home/student** directory if it does not yet exist. If it does exist, empty it of all contents. When **systemd-tmpfiles --clean** is run, remove all files which have not been accessed, changed, or modified in more than one day.

```
L /run/fstablink - root root - /etc/fstab
```

Create the symbolic link **/run/fstablink** pointing to **/etc/fstab**. Never automatically purge this line.

Configuration File Precedence

Configuration files can exist in three places:

- **/etc/tmpfiles.d/*.conf**
- **/run/tmpfiles.d/*.conf**
- **/usr/lib/tmpfiles.d/*.conf**

The files in **/usr/lib/tmpfiles.d/** are provided by the relevant RPM packages, and you should not edit these files. The files under **/run/tmpfiles.d/** are themselves volatile files, normally used by daemons to manage their own runtime temporary files. The files under **/etc/tmpfiles.d/** are meant for administrators to configure custom temporary locations, and to override vendor-provided defaults.

If a file in **/run/tmpfiles.d/** has the same file name as a file in **/usr/lib/tmpfiles.d/**, then the file in **/run/tmpfiles.d/** is used. If a file in **/etc/tmpfiles.d/** has the same file name as a file in either **/run/tmpfiles.d/** or **/usr/lib/tmpfiles.d/**, then the file in **/etc/tmpfiles.d/** is used.

Given these precedence rules, you can easily override vendor-provided settings by *copying* the relevant file to **/etc/tmpfiles.d/**, and then editing it. Working in this fashion ensures that administrator-provided settings can be easily managed from a central configuration management system, and not be overwritten by an update to a package.

NOTE

When testing new or modified configurations, it can be useful to only apply the commands from one configuration file. This can be achieved by specifying the name of the configuration file on the command line.

REFERENCES

systemd-tmpfiles(8), **tmpfiles.d**(5), **stat**(1), **stat**(2), and **systemd.timer**(5) man pages

GUIDED EXERCISE

MANAGING TEMPORARY FILES

In this exercise, you will configure **systemd-tmpfiles** in order to change how quickly it removes temporary files from **/tmp**, and also to periodically purge files from another directory.

OUTCOMES

You should be able to:

- Configure **systemd-tmpfiles** to remove unused temporary files from **/tmp**.
- Configure **systemd-tmpfiles** to periodically purge files from another directory.

BEFORE YOU BEGIN

Log in to workstation as student using student as the password.

On workstation, run **lab scheduling-tempfiles start** to start the exercise. This script creates the necessary files and ensures that the environment is set up correctly.

```
[student@workstation ~]$ lab scheduling-tempfiles start
```

1. From workstation, open an SSH session to servera as student.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

2. Configure **systemd-tmpfiles** to clean the **/tmp** directory so that it does not contain files that that have not been used in the last five days. Ensure that the configuration does not get overwritten by any package update.

   2.1. Use the **sudo -i** command to switch to the root user.

```
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

   2.2. Copy **/usr/lib/tmpfiles.d/tmp.conf** to **/etc/tmpfiles.d/tmp.conf**.

```
[root@servera ~]# cp /usr/lib/tmpfiles.d/tmp.conf /etc/tmpfiles.d/
tmp.conf
```

   2.3. Search for the configuration line in **/etc/tmpfiles.d/tmp.conf** that applies to the **/tmp** directory. Replace the existing age of the temporary files in that configuration line with the new age of **5** days. Remove all the other lines from the

file including the commented ones. You can use the **vim /etc/tmpfiles.d/tmp.conf** command to edit the configuration file. The **/etc/tmpfiles.d/tmp.conf** file should appear as follows:

```
q /tmp 1777 root root 5d
```

The preceding configuration causes **systemd-tmpfiles** to ensure that the directory **/tmp** exists with the octal permissions set to **1777**. Both the owning user and group of **/tmp** must be `root`. The **/tmp** directory must be free from the temporary files which are unused in the last five days.

2.4. Use the **systemd-tmpfiles --clean** command to verify that the **/etc/tmpfiles.d/tmp.conf** file contains the correct configuration.

```
[root@servera ~]# systemd-tmpfiles --clean /etc/tmpfiles.d/tmp.conf
```

Because the preceding command did not return any errors, it confirms that the configuration settings are correct.

3. Add a new configuration that ensures that the **/run/momentary** directory exists with user and group ownership set to `root`. The octal permissions for the directory must be **0700**. The configuration should purge any file in this directory that remains unused in the last 30 seconds.

3.1. Create the file called **/etc/tmpfiles.d/momentary.conf** with the following content. You can use the **vim /etc/tmpfiles.d/momentary.conf** command to create the configuration file.

```
d /run/momentary 0700 root root 30s
```

The preceding configuration causes **systemd-tmpfiles** to ensure that the **/run/momentary** directory exists with its octal permissions set to **0700**. The user and group ownership of **/run/momentary** must be `root`. Any file in this directory that remains unused in the last 30 seconds must be purged.

3.2. Use the **systemd-tmpfiles --create** command to verify that the **/etc/tmpfiles.d/momentary.conf** file contains the appropriate configuration. The command creates the **/run/momentary** directory if it does not exist.

```
[root@servera ~]# systemd-tmpfiles --create /etc/tmpfiles.d/
momentary.conf
```

Because the preceding command did not return any errors, it confirms that the configuration settings are correct.

3.3. Use the **ls** command to verify that the **/run/momentary** directory is created with the appropriate permissions, owner, and group owner.

```
[root@servera ~]# ls -ld /run/momentary
drwx------. 2 root root 40 Mar 21 16:39 /run/momentary
```

Notice that the octal permission set of **/run/momentary** is **0700** and that the user and group ownership are set to `root`.

4. Verify that any file under the **/run/momentary** directory, unused in the last 30 seconds, is removed, based on the **systemd-tmpfiles** configuration for the directory.

4.1. Use the **touch** command to create a file called **/run/momentary/testfile**.

```
[root@servera ~]# touch /run/momentary/testfile
```

4.2. Use the **sleep** command to configure your shell prompt not to return for 30 seconds.

```
[root@servera ~]# sleep 30
```

4.3. After your shell prompt returns, use the **systemd-tmpfiles --clean** command to clean stale files from **/run/momentary**, based on the rule mentioned in **/etc/tmpfiles.d/momentary.conf**.

```
[root@servera ~]# systemd-tmpfiles --clean /etc/tmpfiles.d/
momentary.conf
```

The preceding command removes **/run/momentary/testfile** because the file remained unused for 30 seconds and should have been removed based on the rule mentioned in **/etc/tmpfiles.d/momentary.conf**.

4.4. Use the **ls -l** command to verify that the **/run/momentary/testfile** file does not exist.

```
[root@servera ~]# ls -l /run/momentary/testfile
ls: cannot access '/run/momentary/testfile': No such file or directory
```

4.5. Exit the `root` user's shell to return to the `student` user.

```
[root@servera ~]# exit
logout
[student@servera ~]$
```

4.6. Log off from `servera`.

```
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```

Finish

On `workstation`, run **lab scheduling-tempfiles finish** to complete this exercise. This script deletes the files created throughout the exercise and ensures that the environment is clean.

```
[student@workstation ~]$ lab scheduling-tempfiles finish
```

This concludes the guided exercise.

QUIZ

SCHEDULING FUTURE TASKS

Choose the correct answers to the following questions.

1. **Which command displays all the user jobs that are currently scheduled to run as deferred jobs?**
   a. `atq`
   b. `atrm`
   c. `at -c`
   d. `at --display`

2. **Which command removes the deferred user job that has the job number 5?**
   a. `at -c 5`
   b. `atrm 5`
   c. `at 5`
   d. `at --delete 5`

3. **Which command displays all the recurring user jobs scheduled for the currently logged-in user?**
   a. `crontab -r`
   b. `crontab -l`
   c. `crontab -u`
   d. `crontab -V`

4. **Which job format executes `/usr/local/bin/daily_backup` hourly from 9 a.m. to 6 p.m. on all days from Monday through Friday?**
   a. `00 * * * Mon-Fri /usr/local/bin/daily_backup`
   b. `* */9 * * Mon-Fri /usr/local/bin/daily_backup`
   c. `00 */18 * * * /usr/local/bin/daily_backup`
   d. `00 09-18 * * Mon-Fri /usr/local/bin/daily_backup`

5. **Which directory contains the shell scripts intended to run on a daily basis?**
   a. `/etc/cron.d`
   b. `/etc/cron.hourly`
   c. `/etc/cron.daily`
   d. `/etc/cron.weekly`

6. **Which configuration file defines the settings for the system jobs that run on a daily, weekly, and monthly basis?**
   a. **`/etc/crontab`**
   b. **`/etc/anacrontab`**
   c. **`/etc/inittab`**
   d. **`/etc/sysconfig/crond`**

7. **Which `systemd` unit regularly triggers the cleanup of the temporary files?**
   a. `systemd-tmpfiles-clean.timer`
   b. `systemd-tmpfiles-clean.service`
   c. `dnf-makecache.timer`
   d. `unbound-anchor.timer`

SOLUTION

SCHEDULING FUTURE TASKS

Choose the correct answers to the following questions.

1. **Which command displays all the user jobs that are currently scheduled to run as deferred jobs?**
   a. **`atq`**
   b. `atrm`
   c. `at -c`
   d. `at --display`

2. **Which command removes the deferred user job that has the job number 5?**
   a. `at -c 5`
   b. **`atrm 5`**
   c. `at 5`
   d. `at --delete 5`

3. **Which command displays all the recurring user jobs scheduled for the currently logged-in user?**
   a. `crontab -r`
   b. **`crontab -l`**
   c. `crontab -u`
   d. `crontab -V`

4. **Which job format executes `/usr/local/bin/daily_backup` hourly from 9 a.m. to 6 p.m. on all days from Monday through Friday?**
   a. `00 * * * Mon-Fri /usr/local/bin/daily_backup`
   b. `* */9 * * Mon-Fri /usr/local/bin/daily_backup`
   c. `00 */18 * * * /usr/local/bin/daily_backup`
   d. **`00 09-18 * * Mon-Fri /usr/local/bin/daily_backup`**

5. **Which directory contains the shell scripts intended to run on a daily basis?**
   a. `/etc/cron.d`
   b. `/etc/cron.hourly`
   c. **`/etc/cron.daily`**
   d. `/etc/cron.weekly`

6. **Which configuration file defines the settings for the system jobs that run on a daily, weekly, and monthly basis?**
   a. `/etc/crontab`
   b. **`/etc/anacrontab`**
   c. `/etc/inittab`
   d. `/etc/sysconfig/crond`

7. **Which `systemd` unit regularly triggers the cleanup of the temporary files?**
   a. `systemd-tmpfiles-clean.timer`
   b. `systemd-tmpfiles-clean.service`
   c. `dnf-makecache.timer`
   d. `unbound-anchor.timer`

SUMMARY

In this chapter, you learned:

- Jobs that are scheduled to run once in the future are called deferred jobs or tasks.
- Recurring user jobs execute the user's tasks on a repeating schedule.
- Recurring system jobs accomplish administrative tasks on a repeating schedule that have system-wide impact.
- The `systemd` timer units can execute both the deferred or recurring jobs.

CHAPTER 3

TUNING SYSTEM PERFORMANCE

GOAL

Improve system performance by setting tuning parameters and adjusting scheduling priority of processes.

OBJECTIVES

- Optimize system performance by selecting a tuning profile managed by the tuned daemon.
- Prioritize or de-prioritize specific processes with the nice and renice commands.

SECTIONS

- Adjusting Tuning Profiles (and Guided Exercise)
- Influencing Process Scheduling (and Guided Exercise)

LAB

Tuning System Performance

ADJUSTING TUNING PROFILES

OBJECTIVES

After completing this section, you should be able to optimize system performance by selecting a tuning profile managed by the `tuned` daemon.

TUNING SYSTEMS

System administrators can optimize the performance of a system by adjusting various device settings based on a variety of use case workloads. The `tuned` daemon applies tuning adjustments both statically and dynamically, using tuning profiles that reflect particular workload requirements.

Configuring Static Tuning

The `tuned` daemon applies system settings when the service starts or upon selection of a new tuning profile. Static tuning configures predefined `kernel` parameters in profiles that **`tuned`** applies at runtime. With static tuning, kernel parameters are set for overall performance expectations, and are not adjusted as activity levels change.

Configuring Dynamic Tuning

With dynamic tuning, the `tuned` daemon monitors system activity and adjusts settings depending on runtime behavior changes. Dynamic tuning is continuously adjusting tuning to fit the current workload, starting with the initial settings declared in the chosen tuning profile.

For example, storage devices experience high use during startup and login, but have minimal activity when user workloads consist of using web browsers and email clients. Similarly, CPU and network devices experience activity increases during peak usage throughout a workday. The `tuned` daemon monitors the activity of these components and adjusts parameter settings to maximize performance during high-activity times and reduce settings during low activity. The `tuned` daemon uses performance parameters provided in predefined tuning profiles.

INSTALLING AND ENABLING TUNED

A minimal Red Hat Enterprise Linux 8 installation includes and enables the *tuned* package by default. To install and enable the package manually:

```
[root@host ~]$ yum install tuned
[root@host ~]$ systemctl enable --now tuned
Created symlink /etc/systemd/system/multi-user.target.wants/tuned.service → /usr/
lib/systemd/system/tuned.service.
```

SELECTING A TUNING PROFILE

The ***Tuned*** application provides profiles divided into the following categories:

- Power-saving profiles
- Performance-boosting profiles

The performance-boosting profiles include profiles that focus on the following aspects:

- Low latency for storage and network
- High throughput for storage and network
- Virtual machine performance
- Virtualization host performance

Tuning Profiles Distributed with Red Hat Enterprise Linux 8

TUNED PROFILE	PURPOSE
`balanced`	Ideal for systems that require a compromise between power saving and performance.
`desktop`	Derived from the `balanced` profile. Provides faster response of interactive applications.
`throughput-performance`	Tunes the system for maximum throughput.
`latency-performance`	Ideal for server systems that require low latency at the expense of power consumption.
`network-latency`	Derived from the `latency-performance` profile. It enables additional network tuning parameters to provide low network latency.
`network-throughput`	Derived from the `throughput-performance` profile. Additional network tuning parameters are applied for maximum network throughput.
`powersave`	Tunes the system for maximum power saving.
`oracle`	Optimized for Oracle database loads based on the `throughput-performance` profile.
`virtual-guest`	Tunes the system for maximum performance if it runs on a virtual machine.
`virtual-host`	Tunes the system for maximum performance if it acts as a host for virtual machines.

MANAGING PROFILES FROM THE COMMAND LINE

The **`tuned-adm`** command is used to change settings of the `tuned` daemon. The **`tuned-adm`** command can query current settings, list available profiles, recommend a tuning profile for the system, change profiles directly, or turn off tuning.

A system administrator identifies the currently active tuning profile with **`tuned-adm active`**.

```
[root@host ~]# tuned-adm active
Current active profile: virtual-guest
```

The **`tuned-adm list`** command lists all available tuning profiles, including both built-in profiles and custom tuning profiles created by a system administrator.

```
[root@host ~]# tuned-adm list
Available profiles:
- balanced
- desktop
- latency-performance
- network-latency
- network-throughput
- powersave
- sap
- throughput-performance
- virtual-guest
- virtual-host
Current active profile: virtual-guest
```

Use **`tuned-adm profile`** ***`profilename`*** to switch the active profile to a different one that better matches the system's current tuning requirements.

```
[root@host ~]$ tuned-adm profile throughput-performance
[root@host ~]$ tuned-adm active
Current active profile: throughput-performance
```

The **`tuned-adm`** command can recommend a tuning profile for the system. This mechanism is used to determine the default profile of a system after installation.

```
[root@host ~]$ tuned-adm recommend
virtual-guest
```

NOTE

The **`tuned-adm recommend`** output is based on various system characteristics, including whether the system is a virtual machine and other predefined categories selected during system installation.

To revert the setting changes made by the current profile, either switch to another profile or deactivate the tuned daemon. Turn off `tuned` tuning activity with **`tuned-adm off`**.

```
[root@host ~]$ tuned-adm off
[root@host ~]$ tuned-adm active
No current active profile.
```

MANAGING PROFILES WITH WEB CONSOLE

To manage system performance profiles with Web Console, log in with privileged access. Click the ***Reuse my password for privileged tasks*** option. This permits the user to execute commands, with sudo privileges, that modify system performance profiles.

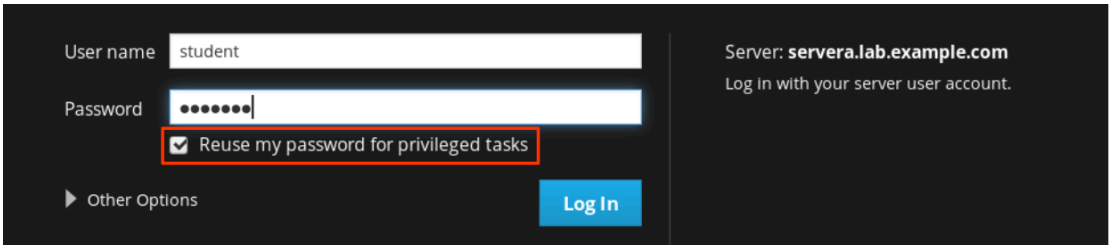

Figure 3.1: Web Console privileged login

As a privileged user, click the ***Systems*** menu option in the left navigation bar. The current active profile is displayed in the ***Performance Profile*** field. To select a different profile, click the active profile link.

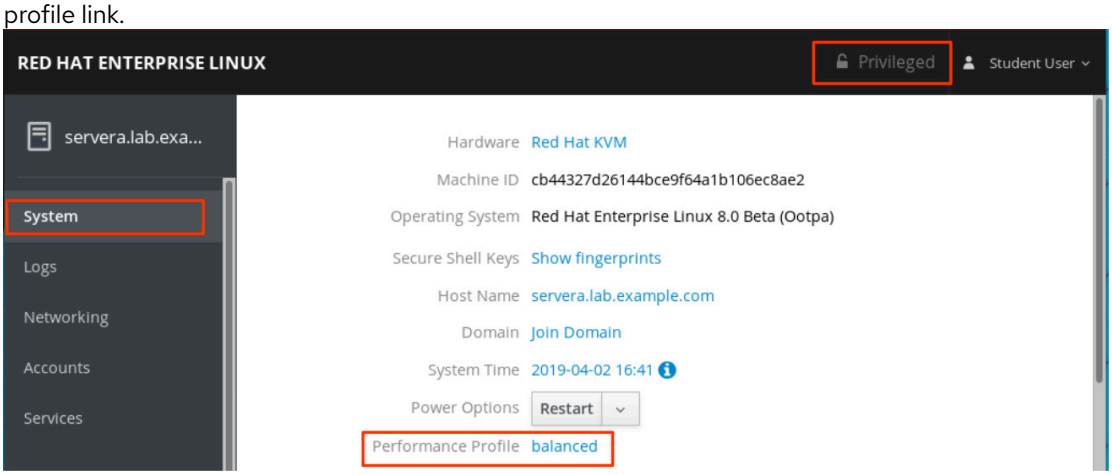

Figure 3.2: Active performance profile

In the ***Change Performance Profile*** user interface, scroll through the profile list to select one that best suits the system purpose.

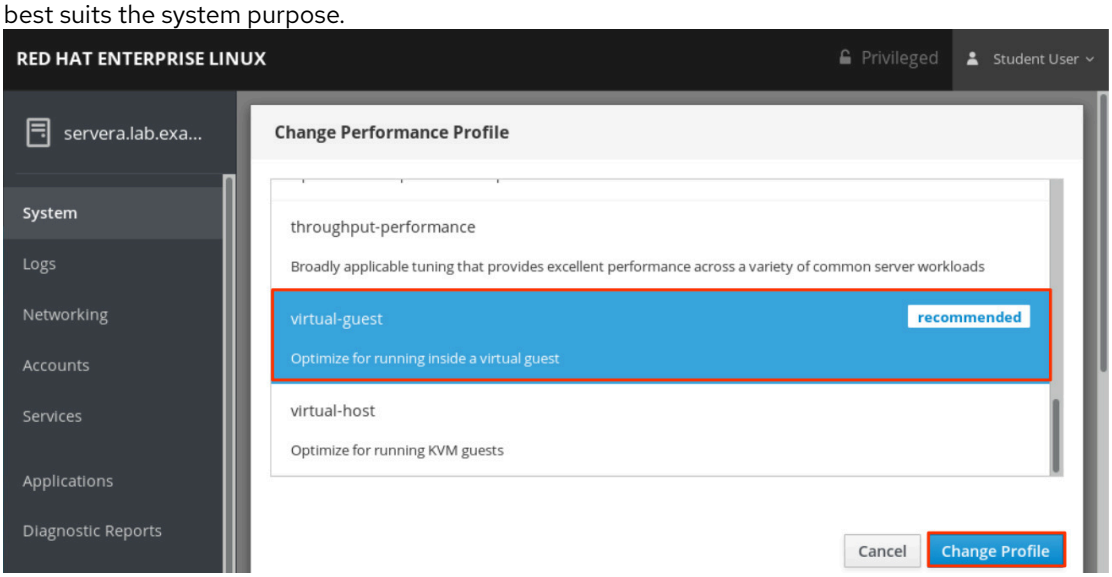

Figure 3.3: Select a preferred performance profile

To verify changes, return to the main ***System*** page and confirm that it displays the active profile in the ***Performance Profile*** field.

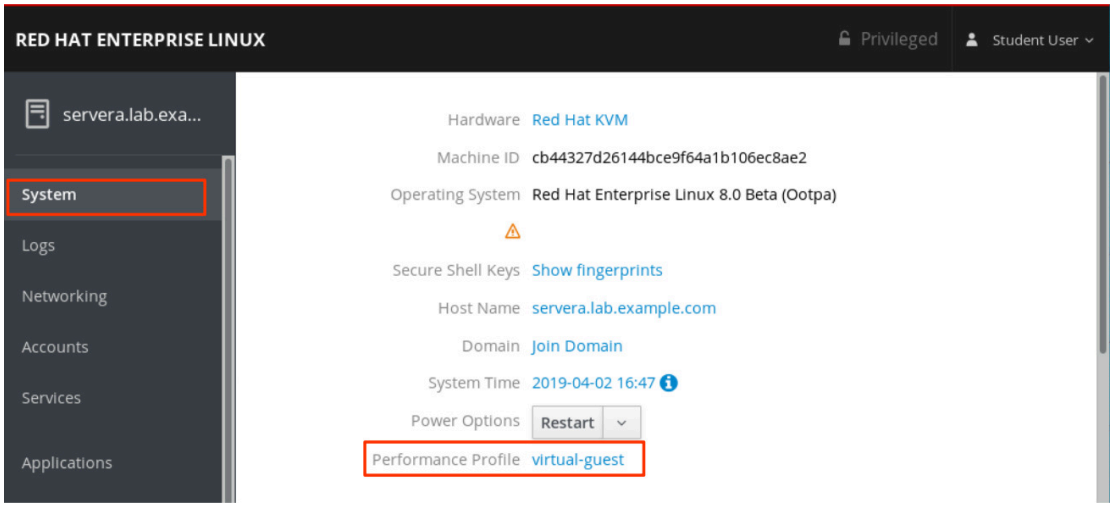

Figure 3.4: Verify active performance profile

REFERENCES

`tuned`(8), **`tuned.conf`**(5), **`tuned-main.conf`**(5) and, **`tuned-adm`**(1) man pages

GUIDED EXERCISE

ADJUSTING TUNING PROFILES

In this exercise, you will tune a server's performance by activating the `tuned` service and applying a tuning profile.

OUTCOMES

You should be able to configure a system to use a tuning profile.

BEFORE YOU BEGIN

Log in as the `student` user on `workstation` using `student` as the password.

From `workstation`, run the **`lab tuning-profiles start`** command. The command runs a start script to determine if the `servera` host is reachable on the network.

```
[student@workstation ~]$ lab tuning-profiles start
```

1. From `workstation`, use SSH to log in to `servera` as the `student` user. The systems are configured to use SSH keys for authentication, therefore a password is not required.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

2. Verify that the *tuned* package is installed, enabled, and started.

   2.1. Use **`yum`** to confirm that the *tuned* package is installed.

```
[student@servera ~]$ yum list tuned
...output omitted...
Installed Packages
tuned.noarch                    2.10.0-15.el8
 @anaconda
```

   2.2. The **`systemctl is-enabled tuned; systemctl is-active tuned`** command displays its enablement and run state.

```
[student@servera ~]$ systemctl is-enabled tuned; systemctl is-active
 tuned
enabled
active
```

3. List the available tuning profiles and identify the active profile. If `sudo` prompts for a password, enter `student` after the prompt.

```
[student@servera ~]$ sudo tuned-adm list
[sudo] password for student: student
Available profiles:
- balanced                    - General non-specialized tuned profile
- desktop                     - Optimize for the desktop use-case
- latency-performance         - Optimize for deterministic performance at the
 cost of
                                increased power consumption
- network-latency             - Optimize for deterministic performance at the
 cost of
                                increased power consumption, focused on low
 latency
                                network performance
- network-throughput          - Optimize for streaming network throughput,
 generally
                                only necessary on older CPUs or 40G+ networks
- powersave                   - Optimize for low power consumption
- throughput-performance      - Broadly applicable tuning that provides excellent
                                performance across a variety of common server
 workloads
- virtual-guest               - Optimize for running inside a virtual guest
- virtual-host                - Optimize for running KVM guests
Current active profile: virtual-guest
```

4. Change the current active tuning profile to `powersave`, then confirm the results. If `sudo` prompts for a password, enter `student` after the prompt.

   4.1. Change the current active tuning profile.

```
[student@servera ~]$ sudo tuned-adm profile powersave
```

   4.2. Confirm that `powersave` is the active tuning profile.

```
[student@servera ~]$ sudo tuned-adm active
Current active profile: powersave
```

5. Exit from `servera`.

```
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```

Finish

On `workstation`, run the **`lab tuning-profiles finish`** script to finish this exercise.

```
[student@workstation ~]$ lab tuning-profiles finish
```

This concludes the guided exercise.

INFLUENCING PROCESS SCHEDULING

OBJECTIVES

After completing this section, you should be able to prioritize or de-prioritize specific processes, with the **`nice`** and **`renice`** commands.

LINUX PROCESS SCHEDULING AND MULTITASKING

Modern computer systems range from low-end systems that have single CPUs that can only execute a single instruction at any instance of time, to high-performing supercomputers with hundreds of `CPUs` each and dozens or even hundreds of processing cores on each `CPU`, allowing the execution of huge numbers of instructions in parallel. All these systems still have one thing in common: the need to run more process threads than they have CPUs.

Linux and other operating systems run more processes than there are processing units using a technique called *time-slicing* or *multitasking*. The operating system *process scheduler* rapidly switches between processes on a single core, giving the impression that there are multiple processes running at the same time.

RELATIVE PRIORITIES

Different processes have different levels of importance. The process scheduler can be configured to use different scheduling policies for different processes. The scheduling policy used for most processes running on a regular system is called `SCHED_OTHER` (also called `SCHED_NORMAL`), but other policies exist for various workload needs.

Since not all processes are equally important, processes running with the `SCHED_NORMAL` policy can be given a relative priority. This priority is called the *nice value* of a process, which are organized as **`40`** different levels of niceness for any process.

The nice level values range from -20 (highest priority) to 19 (lowest priority). By default, processes inherit their nice level from their parent, which is usually 0. Higher nice levels indicate less priority (the process easily gives up its CPU usage), while lower nice levels indicate a higher priority (the process is less inclined to give up the CPU). If there is no contention for resources, for example, when there are fewer active processes than available CPU cores, even processes with a high nice level will still use all available CPU resources they can. However, when there are more processes requesting CPU time than available cores, the processes with a higher nice level will receive less CPU time than those with a lower nice level.

SETTING NICE LEVELS AND PERMISSIONS

Since setting a low nice level on a CPU-hungry process might negatively impact the performance of other processes running on the same system, only the `root` user may *reduce* a process nice level.

Unprivileged users are only permitted to *increase* nice levels on their own processes. They cannot lower the nice levels on their processes, nor can they modify the nice level of other users' processes.

REPORTING NICE LEVELS

Several tools display the nice levels of running processes. Process management tools, such as ***top***, display the nice level by default. Other tools, such as the **ps** command, display nice levels when using the proper options.

Displaying Nice Levels with Top

Use the **top** command to interactively view and manage processes. The default configuration displays two columns of interest about nice levels and priorities. The **NI** column displays the process nice value and the **PR** column displays its scheduled priority. In the **top** interface, the nice level maps to an internal system priority queue as displayed in the following graphic. For example, a nice level of -20 maps to 0 in the **PR** column. A nice level of 19 maps to a priority of 39 in the **PR** column.

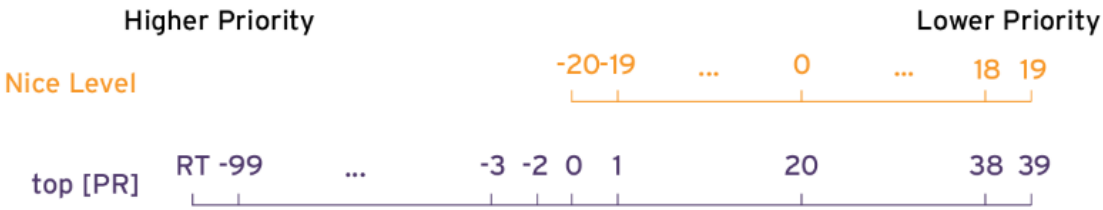

Figure 3.5: Nice levels as reported by top

Displaying Nice Levels from the Command Line

The **ps** command displays process nice levels, but only by including the correct formatting options.

The following **ps** command lists all processes with their PID, process name, nice level, and scheduling class, sorted in descending order by nice level. Processes that display **TS** in the **CLS** scheduling class column, run under the `SCHED_NORMAL` scheduling policy. Processes with a dash (**-**) as their nice level, run under other scheduling policies and are interpreted as a higher priority by the scheduler. Details of the additional scheduling policies are beyond the scope of this course.

```
[user@host ~]$ ps axo pid,comm,nice,cls --sort=-nice
  PID COMMAND          NI CLS
   30 khugepaged       19  TS
   29 ksmd              5  TS
    1 systemd           0  TS
    2 kthreadd          0  TS
    9 ksoftirqd/0       0  TS
   10 rcu_sched         0  TS
   11 migration/0       -  FF
   12 watchdog/0        -  FF
...output omitted...
```

STARTING PROCESSES WITH DIFFERENT NICE LEVELS

During process creation, a process inherit its parent's nice level. When a process is started from the command line, it will inherit its nice level from the shell process where it was started. Typically, this results in new processes running with a nice level of 0.

The following example starts a process from the shell, and displays the process's nice value. Note the use of the PID option in the **ps** to specify the output requested.

```
[user@host ~]$ sha1sum /dev/zero &
[1] 3480
[user@host ~]$ ps -o pid,comm,nice 3480
  PID COMMAND          NI
 3480 sha1sum           0
```

The **nice** command can be used by all users to start commands with a default or higher nice level. Without options, the **nice** command starts a process with the default nice value of 10.

The following example starts the **sha1sum** command as a background job with the default nice level and displays the process's nice level:

```
[user@host ~]$ nice sha1sum /dev/zero &
[1] 3517
[user@host ~]$ ps -o pid,comm,nice 3517
  PID COMMAND          NI
 3517 sha1sum          10
```

Use the **-n** option to apply a user-defined nice level to the starting process. The default is to add 10 to the process' current nice level. The following example starts a command as a background job with a user-defined nice value and displays the process's nice level:

```
[user@host ~]$ nice -n 15 sha1sum &
[1] 3521
[user@host ~]$ ps -o pid,comm,nice 3521
  PID COMMAND          NI
 3521 sha1sum          15
```

IMPORTANT

Unprivileged users may only increase the nice level from its current value, to a maximum of 19. Once increased, unprivileged users cannot reduce the value to return to the previous nice level. The `root` use may reduce the nice level from any current level, to a minimum of -20.

CHANGING THE NICE LEVEL OF AN EXISTING PROCESS

The nice level of an existing process can be changed using the **renice** command. This example uses the PID identifier from the previous example to change from the current nice level of 15 to the desired nice level of 19.

```
[user@host ~]$ renice -n 19 3521
3521 (process ID) old priority 15, new priority 19
```

The **top** command can also be used to change the nice level on a process. From within the **top** interactive interface, press the **r** option to access the **renice** command, followed by the PID to be changed and the new nice level.

REFERENCES

`nice`(1), **`renice`**(1), **`top`**(1), and **`sched_setscheduler`**(2) man pages.

GUIDED EXERCISE

INFLUENCING PROCESS SCHEDULING

In this exercise, you will adjust the scheduling priority of processes with the **nice** and **renice** commands and observe the effects this has on process execution.

OUTCOMES

You should be able to adjust scheduling priorities for processes.

BEFORE YOU BEGIN

Log in as the `student` user on `workstation` using `student` as the password.

From `workstation`, run the **lab tuning-procscheduling start** command. The command runs a start script to determine if the `servera` host is reachable on the network.

```
[student@workstation ~]$ lab tuning-procscheduling start
```

1. From `workstation` use SSH to log in to `servera` as the `student` user. The systems are configured to use SSH keys for authentication, so a password is not required.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

2. Determine the number of CPU cores on `servera` and then start two instances of the **sha1sum /dev/zero &** command for each core.

   2.1. Use **grep** to parse the number of existing virtual processors (CPU cores) from the **/proc/cpuinfo** file.

```
[student@servera ~]$ grep -c '^processor' /proc/cpuinfo
2
```

   2.2. Use a looping command to start multiple instances of the **sha1sum /dev/zero &** command. Start two per virtual processor found in the previous step. In this example, that would be four instances. The PID values in your output will vary from the example.

```
[student@servera ~]$ for i in $(seq 1 4); do sha1sum /dev/zero & done
[1] 2643
[2] 2644
[3] 2645
[4] 2646
```

3. Verify that the background jobs are running for each of the `sha1sum` processes.

```
[student@servera ~]$ jobs
[1]   Running                 sha1sum /dev/zero &
[2]   Running                 sha1sum /dev/zero &
[3]-  Running                 sha1sum /dev/zero &
[4]+  Running                 sha1sum /dev/zero &
```

4. Use the **`ps`** and **`pgrep`** commands to display the percentage of CPU usage for each `sha1sum` process.

```
[student@servera ~]$ ps u $(pgrep sha1sum)
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
student   2643 49.8  0.0 228360  1744 pts/0    R    11:15   6:09 sha1sum /
dev/zero
student   2644 49.8  0.0 228360  1780 pts/0    R    11:15   6:09 sha1sum /
dev/zero
student   2645 49.8  0.0 228360  1748 pts/0    R    11:15   6:09 sha1sum /
dev/zero
student   2646 49.8  0.0 228360  1780 pts/0    R    11:15   6:09 sha1sum /
dev/zero
```

5. Terminate all `sha1sum` processes, then verify that there are no running jobs.

   5.1. Use the **`pkill`** command to terminate all running processes with the name pattern `sha1sum`.

```
[student@servera ~]$ pkill sha1sum
[2]   Terminated              sha1sum /dev/zero
[4]+  Terminated              sha1sum /dev/zero
[1]-  Terminated              sha1sum /dev/zero
[3]+  Terminated              sha1sum /dev/zero
```

   5.2. Verify that there are no running jobs.

```
[student@servera ~]$ jobs
[student@servera ~]$
```

6. Start multiple instances of **`sha1sum /dev/zero &`**, then start one additional instance of **`sha1sum /dev/zero &`** with a nice level of 10. Start at least as many instances as the

system has virtual processors. In this example, 3 regular instances are started, plus another with the higher nice level.

6.1. Use looping to start three instances of **sha1sum /dev/zero &**.

```
[student@servera ~]$ for i in $(seq 1 3); do sha1sum /dev/zero & done
[1] 1947
[2] 1948
[3] 1949
```

6.2. Use the **nice** command to start the fourth instance with a 10 nice level.

```
[student@servera ~]$ nice -n 10 sha1sum /dev/zero &
[4] 1953
```

7. Use the **ps** and **pgrep** commands to display the PID, percentage of CPU usage, nice value, and executable name for each process. The instance with the nice value of 10 should display a lower percentage of CPU usage than the other instances.

```
[student@servera ~]$ ps -o pid,pcpu,nice,comm $(pgrep sha1sum)
  PID %CPU  NI COMMAND
 1947 66.0   0 sha1sum
 1948 65.7   0 sha1sum
 1949 66.1   0 sha1sum
 1953  6.7  10 sha1sum
```

8. Use the **sudo renice** command to lower the nice level of a process from the previous step. Note the PID value from the process instance with the nice level of 10. Use that process PID to lower its nice level to 5.

```
[student@servera ~]$ sudo renice -n 5 1953
[sudo] password for student: student
1953 (process ID) old priority 10, new priority 5
```

9. Repeat the **ps** and **pgrep** commands to redisplay the CPU percent and nice level.

```
[student@servera ~]$ ps -o pid,pcpu,nice,comm $(pgrep sha1sum)
  PID %CPU  NI COMMAND
 1947 63.8   0 sha1sum
 1948 62.8   0 sha1sum
 1949 65.3   0 sha1sum
 1953  9.1   5 sha1sum
```

10. Exit from `servera`.

```
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```

Finish

On `workstation`, run the **`lab tuning-procscheduling finish`** script to finish this exercise.

```
[student@workstation ~]$ lab tuning-procscheduling finish
```

This concludes the guided exercise.

LAB

TUNING SYSTEM PERFORMANCE

PERFORMANCE CHECKLIST

In this lab, you will apply a specific tuning profile and adjust the scheduling priority of an existing process with high CPU usage.

OUTCOMES

You should be able to:

- Activate a specific tuning profile for a computer system.
- Adjust the CPU scheduling priority of a process.

BEFORE YOU BEGIN

Log in as the `student` user on `workstation` using `student` as the password.

From `workstation`, run the **`lab tuning-review start`** command. The command runs a start script to determine whether the `serverb` host is reachable on the network.

```
[student@workstation ~]$ lab tuning-review start
```

1. Change the current tuning profile for `serverb` to `balanced`, a general non-specialized tuned profile.
2. Two processes on `serverb` are consuming a high percentage of CPU usage. Adjust each process's `nice` level to 10 to allow more CPU time for other processes.

Evaluation

On `workstation`, run the **`lab tuning-review grade`** command to confirm success of this lab exercise.

```
[student@workstation ~]$ lab tuning-review grade
```

Finish

On `workstation`, run the **`lab tuning-review finish`** script to complete this exercise.

```
[student@workstation ~]$ lab tuning-review finish
```

This concludes the lab.

SOLUTION

TUNING SYSTEM PERFORMANCE

PERFORMANCE CHECKLIST

In this lab, you will apply a specific tuning profile and adjust the scheduling priority of an existing process with high CPU usage.

OUTCOMES

You should be able to:

- Activate a specific tuning profile for a computer system.
- Adjust the CPU scheduling priority of a process.

BEFORE YOU BEGIN

Log in as the `student` user on `workstation` using `student` as the password.

From `workstation`, run the **`lab tuning-review start`** command. The command runs a start script to determine whether the `serverb` host is reachable on the network.

```
[student@workstation ~]$ lab tuning-review start
```

1. Change the current tuning profile for `serverb` to `balanced`, a general non-specialized tuned profile.

1.1. From `workstation`, open an SSH session to `serverb` as `student` user. The systems are configured to use SSH keys for authentication, so a password is not required.

```
[student@workstation ~]$ ssh student@serverb
...output omitted...
[student@serverb ~]$
```

1.2. Use **`yum`** to confirm that the *tuned* package is installed.

```
[student@serverb ~]$ yum list tuned
...output omitted...
Installed Packages
tuned.noarch                    2.10.0-15.el8
 @anaconda
```

1.3. Use the **`systemctl is-enabled tuned`** command to display the `tuned` service enablement state.

```
[student@serverb ~]$ systemctl is-enabled tuned
enabled
```

1.4. List all available tuning profiles and their descriptions. Note that the current active profile is `virtual-guest`.

```
[student@serverb ~]$ sudo tuned-adm list
[sudo] password for student: student
Available profiles:
- balanced                    - General non-specialized tuned profile
- desktop                     - Optimize for the desktop use-case
- latency-performance         - Optimize for deterministic performance at
 the cost of
                                increased power consumption
- network-latency             - Optimize for deterministic performance at
 the cost of
                                increased power consumption, focused on low
 latency
                                network performance
- network-throughput          - Optimize for streaming network throughput,
 generally
                                only necessary on older CPUs or 40G+
 networks
- powersave                   - Optimize for low power consumption
- throughput-performance      - Broadly applicable tuning that provides
 excellent
                                performance across a variety of common
 server workloads
- virtual-guest               - Optimize for running inside a virtual guest
- virtual-host                - Optimize for running KVM guests
Current active profile: virtual-guest
```

1.5. Change the current active tuning profile to the `balanced` profile.

```
[student@serverb ~]$ sudo tuned-adm profile balanced
```

1.6. List summary information of the current active tuned profile.

Use the **`tuned-adm profile_info`** command to confirm that the active profile is the `balanced` profile.

```
[student@serverb ~]$ sudo tuned-adm profile_info
Profile name:
balanced

Profile summary:
General non-specialized tuned profile
...output omitted...
```

2. Two processes on `serverb` are consuming a high percentage of CPU usage. Adjust each process's `nice` level to 10 to allow more CPU time for other processes.

2.1. Determine the top two CPU consumers on `serverb`. The top CPU consumers are listed last in the command output. CPU percentage values will vary.

```
[student@serverb ~]$ ps aux --sort=pcpu
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME
 COMMAND
...output omitted...
root      2983  100  0.0 228360  1744 ?        R<   21:08
 0:23 md5sum /dev/zero
root      2967  101  0.0 228360  1732 ?        RN   21:08
 0:23 sha1sum /dev/zero
[student@serverb ~]$
```

2.2. Identify the current `nice` level for each of the top two CPU consumers.

```
[student@serverb ~]$ ps -o pid,pcpu,nice,comm $(pgrep sha1sum;pgrep
 md5sum)
  PID %CPU  NI COMMAND
 2967 99.6   2 sha1sum
 2983 99.7  -2 md5sum
```

2.3. Use the **`sudo renice -n 10 2967 2983`** command to adjust the `nice` level for each process to **`10`**. Use PID values identified in the previous command output.

```
[student@serverb ~]$ sudo renice -n 10 2967 2983
[sudo] password for student: student
2967 (process ID) old priority 2, new priority 10
2983 (process ID) old priority -2, new priority 10
```

2.4. Verify that the current `nice` level for each process is 10.

```
[student@serverb ~]$ ps -o pid,pcpu,nice,comm $(pgrep sha1sum;pgrep
 md5sum)
  PID %CPU  NI COMMAND
 2967 99.6  10 sha1sum
 2983 99.7  10 md5sum
```

2.5. Exit from `serverb`.

```
[student@serverb ~]$ exit
logout
Connection to serverb closed.
[student@workstation ~]$
```

Evaluation

On `workstation`, run the **`lab tuning-review grade`** command to confirm success of this lab exercise.

```
[student@workstation ~]$ lab tuning-review grade
```

Finish

On `workstation`, run the **`lab tuning-review finish`** script to complete this exercise.

```
[student@workstation ~]$ lab tuning-review finish
```

This concludes the lab.

SUMMARY

In this chapter, you learned:

- The `tuned` service automatically modifies device settings to meet specific system needs based on a pre-defined selected tuning profile.
- To revert all changes made to system settings by a selected profile, either switch to another profile or deactivate the `tuned` service.
- The system assigns a relative priority to a process to determine its CPU access. This priority is called the **`nice`** value of a process.
- The **`nice`** command assigns a priority to a process when it starts. The **`renice`** command modifies the priority of a running process.

CHAPTER 4

CONTROLLING ACCESS TO FILES WITH ACLS

GOAL

Interpret and set Access Control Lists (ACLs) on files to handle situations requiring complex user and group access permissions.

OBJECTIVES

- Describe use cases for ACLs, identify files that have ACLs set, and interpret the effect of those ACLs.
- Set and remove ACLs on files and define default ACLs automatically set by a directory on newly created files.

SECTIONS

- Interpreting File ACLs (and Guided Exercise)
- Securing Files with ACLs (and Guided Exercise)

LAB

Controlling Access to Files with ACLs

INTERPRETING FILE ACLS

OBJECTIVES

After completing this section, you should be able to:

- Describe ACLs and file-system mount options.
- View and interpret ACLs with **`ls`** and **`getfacl`**.
- Describe the ACL mask and ACL permission precedence.
- Identify where Red Hat Enterprise Linux uses ACLs by default.

ACCESS CONTROL LIST CONCEPTS

Standard Linux file permissions are satisfactory when files are used by only a single owner, and a single designated group of people. However, some use cases require that files are accessed with different file permission sets by multiple named users and groups. *Access Control Lists (ACLs)* provide this function.

With ACLs, you can grant permissions to multiple users and groups, identified by user name, group name, UID, or GID, using the same permission flags used with regular file permissions: read, write, and execute. These additional users and groups, beyond the file owner and the file's group affiliation, are called *named users* and *named groups* respectively, because they are named not in a long listing, but rather within an ACL.

Users can set ACLs on files and directories that they own. Privileged users, assigned the **`CAP_FOWNER`** Linux capability, can set ACLs on any file or directory. New files and subdirectories automatically inherit ACL settings from the parent directory's default ACL, if they are set. Similar to normal file access rules, the parent directory hierarchy needs at least the *other* search (execute) permission set to enable named users and named groups to have access.

File-system ACL Support

File systems need to be mounted with ACL support enabled. XFS file systems have built-in ACL support. Other file systems, such as ext3 or ext4 created on Red Hat Enterprise Linux 8, have the **`acl`** option enabled by default, although on earlier versions you should confirm that ACL support is enabled. To enable file-system ACL support, use the ACL option with the **`mount`** command or in the file system's entry in **`/etc/fstab`** configuration file.

VIEWING AND INTERPRETING ACL PERMISSIONS

The **`ls -l`** command only outputs minimal ACL setting details:

```
[user@host content]$ ls -l reports.txt
-rwxrw----+ 1 user operators 130 Mar 19 23:56 reports.txt
```

The plus sign (+) at the end of the 10-character permission string indicates that an extended ACL structure with entries exists on this file.

user:
Shows the *user* ACL settings, which are the same as the standard *user* file settings; **rwx**.

group:
Shows the current ACL *mask* settings, not the *group owner* settings; **rw**.

other:
Shows the *other* ACL settings, which are the same as the standard *other* file settings; no access.

IMPORTANT
Changing group permissions on a file with an ACL by using **chmod** does not change the group owner permissions, but does change the ACL mask. Use **setfacl -m g::*perms file*** if the intent is to update the file's group owner permissions.

View File ACLs

To display ACL settings on a file, use **getfacl *file***:

```
[user@host content]$ getfacl reports.txt
# file: reports.txt
# owner: user
# group: operators
user::rwx
user:consultant3:---
user:1005:rwx       #effective:rw-
group::rwx          #effective:rw-
group:consultant1:r--
group:2210:rwx      #effective:rw-
mask::rw-
other::---
```

Review each section of the previous example:

Commented entries:

```
# file: reports.txt
# owner: user
# group: operators
```

The first three lines are comments that identify the file name, owner (**user**), and group owner (**operators**). If there are any additional file flags, such as **setuid** or **setgid**, then a fourth comment line will appear showing which flags are set.

User entries:

```
user::rwx                          1
user:consultant3:---                   2
user:1005:rwx       #effective:rw-     3
```

1. File owner permissions. **user** has **rwx**.

2. Named user permissions. One entry for each named user associated with this file. `consultant3` has *no* permissions.
3. Named user permissions. UID `1005` has **`rwx`**, but the mask limits the effective permissions to **`rw`** only.

Group entries:

```
group::rwx          #effective:rw-   1
group:consultant1:r--                2
group:2210:rwx      #effective:rw-   3
```

1. Group owner permissions. `operators` has **`rwx`**, but the mask limits the effective permissions to **`rw`** only.
2. Named group permissions. One entry for each named group associated with this file. `consultant1` has **`r`** only.
3. Named group permissions. GID `2210` has **`rwx`**, but the mask limits the effective permissions to **`rw`** only.

Mask entry:

```
mask::rw-
```

Mask settings show the maximum permissions possible for all named users, the group owner, and named groups. UID `1005`, `operators`, and GID `2210` cannot execute this file, even though each entry has the execute permission set.

Other entry:

```
other::---
```

Other or "world" permissions. All other UIDs and GIDs have NO permissions.

Viewing Directory ACLs

To display ACL settings on a directory, use the **`getfacl`** ***`directory`*** command:

```
[user@host content]$ getfacl .
# file: .
# owner: user
# group: operators
# flags: -s-
user::rwx
user:consultant3:---
user:1005:rwx
group::rwx
group:consultant1:r-x
group:2210:rwx
mask::rwx
other::---
default:user::rwx
default:user:consultant3:---
default:group::rwx
```

```
default:group:consultant1:r-x
default:mask::rwx
default:other::---
```

Review each section of the previous example:

Opening comment entries:

```
# file: .
# owner: user
# group: operators
# flags: -s-
```

The first three lines are comments that identify the directory name, owner (`user`), and group owner (`operators`). If there are any additional directory flags (**`setuid`**, **`setgid`**, **`sticky`**), then a fourth comment line shows which flags are set; in this case, **`setgid`**.

Standard ACL entries:

```
user::rwx
user:consultant3:---
user:1005:rwx
group::rwx
group:consultant1:r-x
group:2210:rwx
mask::rwx
other::---
```

The ACL permissions on this directory are the same as the file example shown earlier, but apply to the directory. The key difference is the inclusion of the execute permission on these entries (when appropriate) to allow directory search permission.

Default user entries:

```
default:user::rwx                    (1)
default:user:consultant3:---         (2)
```

1. Default file owner ACL permissions. The file owner will get **`rwx`**, read/write on new files and execute on new subdirectories.
2. Default named user ACL permissions. One entry for each named user who will automatically get the default ACL applied to new files or subdirectories. `consultant3` always defaults to *no* permissions.

Default group entries:

```
default:group::rwx                   (1)
default:group:consultant1:r-x        (2)
```

1. Default group owner ACL permissions. The file group owner will get **`rwx`**, read/write on new files and execute on new subdirectories.

2. Default named group ACL permissions. One entry for each named group which will automatically get the default ACL. `consultant1` will get **`rx`**, read-only on new files, and execute on new subdirectories.

Default ACL mask entry:

```
default:mask::rwx
```

Default mask settings show the initial maximum permissions possible for all new files or directories created that have named user ACLs, the group owner ACL, or named group ACLs: read and write for new files and execute permission on new subdirectories. New files never get execute permission.

Default other entry:

```
default:other::---
```

Default *other* or "world" permissions. All other UIDs and GIDs have *no* permissions to new files or new subdirectories.

The **`default`** entries in the previous example do not include the named user (UID **`1005`**) and named group (GID **`2210`**); consequently, they will not automatically get initial ACL entries added for them to any new files or new subdirectories. This effectively limits them to files and subdirectories that they already have ACLs on, or if the relevant file owner adds the ACL later using **`setfacl`**. They can still create their own files and subdirectories.

> **NOTE**
> The output from **`getfacl`** command can be used as input to **`setfacl`** for restoring ACLs, or for copying ACLs from a source file or directory and save them into a new file. For example, to restore ACLs from a backup, use **`getfacl -R /dir1 > file1`** to generate a recursive ACL output dump file for the directory and its contents. The output can then be used for recovery of the original ACLs, passing the saved output as input to the **`setfacl`** command. For example, to perform a bulk update of the same directory in the current path use the following command: **`setfacl --set-file=file1`**

The ACL Mask

The ACL mask defines the maximum permissions that you can grant to *named users*, the *group owner*, and *named groups*. It does not restrict the permissions of the *file owner* or `other` users. All files and directories that implement ACLs will have an ACL mask.

The mask can be viewed with **`getfacl`** and explicitly set with **`setfacl`**. It will be calculated and added automatically if it is not explicitly set, but it could also be inherited from a parent directory default mask setting. By default, the mask is recalculated whenever any of the affected ACLs are added, modified, or deleted.

ACL Permission Precedence

When determining whether a process (a running program) can access a file, file permissions and ACLs are applied as follows:

- If the process is running as the user that owns the file, then the file's user ACL permissions apply.
- If the process is running as a user that is listed in a named user ACL entry, then the named user ACL permissions apply (as long as it is permitted by the mask).
- If the process is running as a group that matches the group owner of the file, or as a group with an explicitly named group ACL entry, then the matching ACL permissions apply (as long as it is permitted by the mask).
- Otherwise, the file's *other* ACL permissions apply.

EXAMPLES OF ACL USE BY THE OPERATING SYSTEM

Red Hat Enterprise Linux has examples that demonstrate typical ACL use for extended permission requirements.

ACLs on Systemd Journal Files

`systemd-journald` uses ACL entries to allow read access to the **`/run/log/journal/cb44...8ae2/system.journal`** file to the `adm` and `wheel` groups. This ACL allows the members of the `adm` and `wheel` groups to have read access to the logs managed by **`journalctl`** without having the need to give special permissions to the privileged content inside **`/var/log/`**, like **`messages`**, **`secure`** or **`audit`**.

Due to the `systemd-journald` configuration, the parent folder of the **`system.journal`** file can change, but the `systemd-journald` applies ACLs to the new folder and file automatically.

> **NOTE**
> System administrators should set an ACL on the **`/var/log/journal/`** folder when `systemd-journald` is configured to use persistent storage.

```
[user@host ]$ getfacl /run/log/journal/cb44...8ae2/system.journal
getfacl: Removing leading '/' from absolute path names
# file: run/log/journal/cb44...8ae2/system.journal
# owner: root
# group: systemd-journal
user::rw-
group::r--
group:adm:r--
group:wheel:r--
mask::r--
other::---
```

ACL on Systemd Managed Devices

`systemd-udev` uses a set of `udev` rules that enable the `uaccess` tag to some devices, such as CD/DVD players or writers, USB storage devices, sound cards, and many others. The previous mentioned `udev` rules sets ACLs on those devices to allow users logged in to a graphical user interface (for example `gdm`) to have full control of those devices.

The ACLs will remain active until the user logs out of the GUI. The next user who logs in to the GUI will have a new ACL applied for the required devices.

In the following example, you can see the `user` has an ACL entry with **`rw`** permissions applied to the **`/dev/sr0`** device that is a CD/DVD drive.

```
[user@host ]$ getfacl /dev/sr0
getfacl: Removing leading '/' from absolute path names
# file: dev/sr0
# owner: root
# group: cdrom
user::rw-
user:group:rw-
group::rw-
mask::rw-
other::---
```

REFERENCES

`acl`(5), **`getfacl`**(1), **`journald.conf`**(5), **`ls`**(1), **`systemd-journald`**(8) and **`systemd-udevd`**(8) man pages

QUIZ

INTERPRETING FILE ACLS

Match the following items to their counterparts in the table.

default:m::rx /directory

default:user:mary:rx /directory

g::rw /directory

g::rw file

getfacl /directory

group:hug:rwx /directory

user::rx file

user:mary:rx file

DESCRIPTION	*ACL OPERATION*
Display the ACL on a directory.	
Named user with read and execute permissions for a file.	
File owner with read and execute permissions for a file.	
Read and write permissions for a directory granted to the directory group owner.	
Read and write permissions for a file granted to the file group owner.	
Read, write, and execute permissions for a directory granted to a named group.	
Read and execute permissions set as the default mask.	
Named user granted initial read permission for new files, and read and execute permissions for new subdirectories.	

SOLUTION

INTERPRETING FILE ACLS

Match the following items to their counterparts in the table.

DESCRIPTION	*ACL OPERATION*
Display the ACL on a directory.	`getfacl /directory`
Named user with read and execute permissions for a file.	`user:mary:rx file`
File owner with read and execute permissions for a file.	`user::rx file`
Read and write permissions for a directory granted to the directory group owner.	`g::rw /directory`
Read and write permissions for a file granted to the file group owner.	`g::rw file`
Read, write, and execute permissions for a directory granted to a named group.	`group:hug:rwx /directory`
Read and execute permissions set as the default mask.	`default:m::rx /directory`
Named user granted initial read permission for new files, and read and execute permissions for new subdirectories.	`default:user:mary:rx /directory`

SECURING FILES WITH ACLS

OBJECTIVES

After completing this section, you should be able to:

- Change regular ACL file permissions using **setfacl**.
- Control default ACL file permissions for new files and directories.

CHANGING ACL FILE PERMISSIONS

Use **setfacl** to add, modify, or remove standard ACLs on files and directories.

ACLs use the normal file system representation of permissions, "**r**" for read permission, "**w**" for write permission, and "**x**" for execute permission. A "**-**" (dash) indicates that the relevant permission is absent. When (recursively) setting ACLs, an uppercase "**X**" can be used to indicate that execute permission should only be set on directories and not regular files, unless the file already has the relevant execute permission. This is the same behavior as **chmod**.

Adding or Modifying ACLs

ACLs can be set via the command-line using the **-m** option, or passed in via a file using the **-M** option (use "**-**" (dash) instead of a file name for **stdin**). These two options are the "modify" options; they add new ACL entries or replace specific existing ACL entries on a file or directory. Any other existing ACL entries on the file or directory remain untouched.

> **NOTE**
> Use the **--set** or **--set-file** options to completely replace the ACL settings on a file.

When first defining an ACL on a file, if the add operation does not include settings for the *file owner*, *group owner*, or *other* permissions, then these will be set based on the current standard file permissions (these are also known as the *base* ACL entries and cannot be deleted), and a new *mask* value will be calculated and added as well.

To add or modify a *user* or *named user* ACL:

```
[user@host ~]$ setfacl -m u:name:rX file
```

If *name* is left blank, then it applies to the *file owner*, otherwise *name* can be a username or UID value. In this example, the permissions granted would be read-only, and if already set, execute (unless *file* was a directory, in which case the directory would get the execute permission set to allow directory search).

ACL *file owner* and standard *file owner* permissions are equivalent; consequently, using **chmod** on the *file owner* permissions is equivalent to using **setfacl** on the *file owner* permissions. **chmod** has no effect on named users.

To add or modify a *group* or *named group* ACL:

```
[user@host ~]$ setfacl -m g:name:rw file
```

This follows the same pattern for adding or modifying a user ACL entry. If *name* is left blank, then it applies to the *group owner*. Otherwise, specify a group name or GID value for a *named group*. The permissions would be read and write in this example.

chmod has no effect on any group permissions for files with ACL settings, but it updates the ACL mask.

To add or modify the *other* ACL:

```
[user@host ~]$ setfacl -m o::- file
```

other only accepts permission settings. Typical permission settings for others are: no permissions at all, set with a dash (**-**); and read-only permissions set as usual with **r**. Of course, you can set any of the standard permissions.

ACL *other* and standard *other* permissions are equivalent, so using **chmod** on the *other* permissions is equivalent to using **setfacl** on the *other* permissions.

You can add multiple entries with the same command; use a comma-separated list of entries:

```
[user@host ~]$ setfacl -m u::rwx,g:consultants:rX,o::- file
```

This sets the *file owner* to read, write, and execute, sets the named group **consultants** to read-only and conditional execute, and restricts all `other` users to *no* permissions. The *group owner* maintains existing file or ACL permissions and other "named" entries remain unchanged.

Using `getfacl` as Input

You can use the output from **getfacl** as input to **setfacl**:

```
[user@host ~]$ getfacl file-A | setfacl --set-file=- file-B
```

The **--set-file** option accepts input from a file or from *stdin*. The dash character (-) specifies the use of *stdin*. In this case, *file-B* will have the same ACL settings as *file-A*.

Setting an Explicit ACL Mask

You can set an ACL mask explicitly on a file or directory to limit the maximum effective permissions for named users, the group owner, and named groups. This restricts any existing permissions that exceed the mask, but does not affect permissions that are less permissive than the mask.

```
[user@host ~]$ setfacl -m m::r file
```

This adds a mask value that restricts any *named users*, the *group owner*, and any *named groups* to read-only permission, regardless of their existing settings. The *file owner* and `other` users are not impacted by the mask setting.

getfacl shows an **effective** comment beside entries that are restricted by a mask setting.

IMPORTANT

By default, each time one of the impacted ACL settings (named users, group owner, or named groups) is modified or deleted, the ACL mask is recalculated, potentially resetting a previous explicit mask setting.

To avoid the mask recalculation, use the **-n** option or include a mask setting (**-m m::*perms***) with any **setfacl** operation that modifies mask-affected ACL settings.

Recursive ACL Modifications

When setting an ACL on a directory, use the **-R** option to apply the ACL recursively. Remember that you are likely to want to use the "**X**" (capital X) permission with recursion so that files with the execute permission set retain the setting and directories get the execute permission set to allow directory search. It is considered good practice to also use the uppercase "**X**" when non-recursively setting ACLs because it prevents administrators from accidentally adding execute permissions to a regular file.

```
[user@host ~]$ setfacl -R -m u:name:rX directory
```

This adds the user *name* to the *directory* directory and all existing files and subdirectories, setting read-only and conditional execute permissions.

Deleting ACLs

Deleting specific ACL entries follows the same basic format as the modify operation, except that "*:perms*" is not specified.

```
[user@host ~]$ setfacl -x u:name,g:name file
```

This removes only the named user and the named group from the file or directory ACL. Any other existing ACL entries remain active.

You can include both the delete (**-x**) and modify (**-m**) operations in the same **setfacl** operation.

The mask can only be deleted if there are no other ACLs set (excluding the *base* ACL which cannot be deleted), so it must be deleted last. The file will no longer have any ACLs and **ls -l** will not show the plus sign (+) next to the permissions string. Alternatively, to delete *all* ACL entries on a file or directory (including *default* ACL on directories), use the following command:

```
[user@host ~]$ setfacl -b file
```

CONTROLLING DEFAULT ACL FILE PERMISSIONS

To ensure that files and directories created within a directory inherit certain ACLs, use the *default* ACL on a directory. You can set a *default* ACL and any of the standard ACL settings, including a default mask.

The directory itself still requires standard ACLs for access control because the *default* ACLs do not implement access control for the directory; they only provide ACL permission inheritance support. For example:

```
[user@host ~]$ setfacl -m d:u:name:rx directory
```

This adds a default named user (**d:u:*name***) with read-only permission and execute permission on subdirectories.

The **setfacl** command for adding a *default* ACL for each of the ACL types is exactly the same as for standard ACLs, but prefaced with **d:**. Alternatively, use the **-d** option on the command line.

IMPORTANT

When setting *default* ACLs on a directory, ensure that users will be able to access the contents of new subdirectories created in it by including the execute permission on the *default* ACL.

Users will not automatically get the execute permission set on newly created regular files because unlike new directories, the ACL mask of a new regular file is **rw-**.

NOTE

New files and new subdirectories continue to get their owner UID and primary group GID values set from the creating user, except when the parent directory **setgid** flag is enabled, in which case the primary group GID is the same as the parent directory GID.

Deleting Default ACL Entries

Delete a *default* ACL the same way that you delete a standard ACL, prefacing with **d:**, or use the **-d** option.

```
[user@host ~]$ setfacl -x d:u:name directory
```

This removes the `default` ACL entry that was added in the previous example.

To delete all *default* ACL entries on a directory, use **setfacl -k *directory***.

REFERENCES

acl(5), **setfacl**(1), and **getfacl**(1) man pages

GUIDED EXERCISE

SECURING FILES WITH ACLS

In this exercise, you will use ACL entries to grant access to a directory for a group and deny access for a user, set the default ACL on a directory, and confirm that new files created in that directory inherit the default ACL.

OUTCOMES

You should be able to:

- Use ACL entries to grant access to a group, and deny access to one of its members.
- Verify that the existing files and directories reflect the new ACL permissions.
- Set the default ACL on a directory, and confirm that new files and directories inherit its configuration.

BEFORE YOU BEGIN

Log in to `workstation` as `student` using `student` as the password.

On `workstation`, run the **`lab acl-secure start`** command. This command runs a start script that determines if the `servera` machine is reachable on the network. It also creates the users, groups, directories, and files used in this exercise.

```
[student@workstation ~]$ lab acl-secure start
```

Operators and Consultants are members of an IT support company. They need to start sharing information. `servera` contains a properly configured share directory located at **`/shares/content`** that hosts files.

Currently, only members of the `operators` group have access to this directory, but members of the `consultants` group need full access to this directory.

The `consultant1` user is a member of the `consultants` group but has caused problems on many occasions, so this user should not have access to the directory.

Your task is to add appropriate ACL entries to the directory and its contents so that members of the `consultants` group have full access, but deny the `consultant1` user any access. Make sure that future files and directories stored in **`/shares/content`** get appropriate ACL entries applied.

Important information:

- The `sysadmin1` and `operator1` users are members of the `operators` group.
- The `consultant1` and `consultant2` users are members of the `consultants` group.
- The **`/shares/content`** directory contains a subdirectory called **`server-info`** and numerous files to test the ACL. Also, the **`/shares/content`** directory contains an executable script called **`loadvg.sh`** that you can use for testing.

- The `sysadmin1`, `operator1`, `consultant1`, and `consultant2` users have their passwords set to `redhat`.
- All changes should occur to the **`/shares/content`** directory and its files; do not adjust the **`/shares`** directory.

1. Log in to `servera` and switch to the `root` user.

   1.1. Use the **`ssh`** command to log in to `servera` as the `student` user. The systems are configured to use SSH keys for authentication, therefore a password is not required.

   ```
   [student@workstation ~]$ ssh student@servera
   ...output omitted...
   [student@servera ~]$
   ```

   1.2. Use the **`sudo -i`** command to switch to the `root` user. The password for the `student` user is `student`.

   ```
   [student@servera ~]$ sudo -i
   [sudo] password for student: student
   [root@servera ~]#
   ```

2. Add the named ACL to the **`/shares/content`** directory and all of its content.

   2.1. Use **`setfacl`** to recursively update the **`/shares/content`** directory, granting the `consultants` group read, write, and conditional execute permissions.

   ```
   [root@servera ~]# setfacl -Rm g:consultants:rwX /shares/content
   ```

   The **`-R`** option means recursive, the **`-m`** option means modify/add, **`rwX`** means apply read, write, and conditional execute permissions.

   2.2. Use **`setfacl`** to recursively update the **`/shares/content`** directory, denying the `consultant1` user from the `consultants` group any access.

   ```
   [root@servera ~]# setfacl -Rm u:consultant1:- /shares/content
   ```

   The **`-R`** option means recursive, the **`-m`** option means modify/add, **`-`** means give no access.

3. Add the named ACL as the *default* ACL to support future file and directory additions.

   3.1. Use **`setfacl`** to add a default access rule for the `consultants` group. Grant read, write, and execute permissions on the **`content`** directory.

   ```
   [root@servera ~]# setfacl -m d:g:consultants:rwx /shares/content
   ```

   The **`-m`** options means modify/add, **`d:g`** means default group, **`rwx`** means apply read/write/execute permissions (needed for proper subdirectory creation and access)

3.2. Use **`setfacl`** to add a default access rule for the `consultant1` user. Deny all access to the **`content`** directory.

```
[root@servera ~]# setfacl -m d:u:consultant1:- /shares/content
```

The **`-m`** option means modify/add, **`d:u`** means default user, **`-`** means no permissions

4. Verify your ACL changes.

`consultant2` should be able to read any file and create a new directory with a new file in it.

`consultant1` should not be able to read, write, or execute any file; this includes being unable to list the directory contents.

Use **`su - user`** to switch to your test users. Use **`exit`** or **`Ctrl`**+**`D`** to leave the test user shell.

```
[root@servera ~]# exit
[student@servera ~]$ su - consultant2
Password: redhat
[consultant2@servera ~]$ cd /shares/content/
```

4.1. Use **`cat`** to check that `consultant2` can read a file.

```
[consultant2@servera content]$ cat serverb-loadavg.txt
##################################################
serverb.lab.example.com
##################################################
Wed Mar 25 15:25:19 EDT 2019
##################################################
ldavg 0.18, 0.06, 0.05
##################################################
```

4.2. Use the **`loadavg.sh`** script to check that `consultant2` can execute a file.

```
[consultant2@servera content]$ ./loadavg.sh
ldavg 0.00, 0.00, 0.04
```

4.3. Create a directory called **`reports`**.

Use **`echo`** to create a file with some content, name the file **`test.txt`**, and place it in the new directory.

Switch back to `student` when you are finished.

```
[consultant2@servera content]$ mkdir reports
[consultant2@servera content]$ echo "TEST REPORT" > reports/test.txt
[consultant2@servera content]$ exit
logout
[student@servera ~]$
```

4.4. Log in as the `consultant1` user. Use **`cd`** to try and change into the directory as `consultant1`, and also try **`ls`** to list the directory. Both commands should fail with **`Permission denied`**.

Try one or more of the commands that `consultant2` used, but as `consultant1`, to further verify the lack of access. Use the full path, **`/shares/content`**, because you cannot use **`cd`** to change into the directory.

Switch back to `student` when you are finished testing `consultant1`.

```
[student@servera ~]$ su - consultant1
Password: redhat
[consultant1@servera ~]$ cd /shares/content/
-bash: cd: /shares/content/: Permission denied
[consultant1@servera ~]$ ls /shares/content/
ls: cannot open directory '/shares/content/': Permission denied
[consultant1@servera ~]$ cat /shares/content/serverb-loadavg.txt
cat: /shares/content/serverb-loadavg.txt: Permission denied
[consultant1@servera ~]$ exit
logout
```

4.5. Log in as the `sysadmin1` user. Use **`getfacl`** to see all the ACL entries on **`/shares/content`** and the ACL entries on **`/shares/content/reports`**.

Switch back to `student` when you are finished testing `consultant1`.

```
[student@servera ~]$ su - sysadmin1
Password: redhat
[sysadmin1@servera ~]$ getfacl /shares/content
getfacl: Removing leading '/' from absolute path names
# file: shares/content/
# owner: root
# group: operators
# flags: -s-
user::rwx
user:consultant1:---
group::rwx
group:consultants:rwx
mask::rwx
other::---
default:user::rwx
default:user:consultant1:---
default:group::rwx
default:group:consultants:rwx
default:mask::rwx
default:other::---

[sysadmin1@servera ~]$ getfacl /shares/content/reports
getfacl: Removing leading '/' from absolute path names
# file: shares/content/reports
# owner: consultant2
# group: operators
# flags: -s-
user::rwx
user:consultant1:---
group::rwx
group:consultants:rwx
mask::rwx
other::---
default:user::rwx
default:user:consultant1:---
default:group::rwx
default:group:consultants:rwx
```

```
default:mask::rwx
default:other::---

[sysadmin1@servera ~]$ exit
logout
```

4.6. Log off from `servera`.

```
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```

Finish

On `workstation`, run the **`lab acl-secure finish`** script to complete this exercise.

```
[student@workstation ~]$ lab acl-secure finish
```

This concludes the guided exercise.

LAB

CONTROLLING ACCESS TO FILES WITH ACLS

PERFORMANCE CHECKLIST

In this lab, you will set up a collaborative directory for users in two groups, combining the set-GID permission and default ACL entries to provide correct access permissions.

OUTCOMES

You should be able to:

- Configure set-GID permission on a folder, to inherit group ownership on files and folders inside.
- Configure ACL entries to allow or deny read/write/execute permissions to users and groups on files and directories.
- Configure default ACL to get the right ACL and file permissions automatically, on new files and directories.

BEFORE YOU BEGIN

Log in to `workstation` as `student` using `student` as the password.

On `workstation`, run the **`lab acl-review start`** command. This command runs a start script that determines if the `serverb` machine is reachable on the network. It also creates the users, groups, directories and files used in this exercise.

```
[student@workstation ~]$ lab acl-review start
```

A stock finance agency is setting up a collaborative share directory to hold case files, which members of the `managers` group will have read and write permissions on.

The co-founder of the agency, `manager1`, has decided that members of the `contractors` group should also be able to read and write to the share directory. However, `manager1` does not trust the `contractor3` user (a member of the `contractors` group), and as such, `contractor3` should have access to the directory restricted to read-only.

`manager1` has created the users and groups, and has started the process of setting up the share directory, copying in some templates files. Because `manager1` has been too busy, it falls to you to finish the job.

Your task is to complete the setup of the share directory. The directory and all of its contents should be owned by the `managers` group, with the files updated to read and write for the owner and group (`managers`). Other users should have no permissions. You also need to provide read and write permissions for the `contractors` group, with the exception of `contractor3`, who only gets read permissions. Make sure your setup applies to existing and future files.

Important information:

- Share directory: **/shares/cases** on `serverb`.
- The `manager1` and `manager2` users are members of the `managers` group.
- The `contractor1`, `contractor2`, and `contractor3` users are members of the `contractors` group.
- Two files exist in the directory: **shortlist.txt** and **backlog.txt**.
- All five user passwords are `redhat`.
- All changes should occur to the **/shares/cases** directory and its files; do not adjust the **/shares** directory.

1. The **cases** directory and its contents should belong to the `managers` group. New files added to the **cases** directory should automatically belong to the `managers` group. The user and group owners for the existing files should have read and write permission, and other users should have no permission at all.

> **NOTE**
> Hint: Do not use **setfacl**.

2. Add ACL entries to the **cases** directory (and its contents) that allow members of the `contractors` group to have read/write access on the files and execute permission on the directory. Restrict the `contractor3` user to read access on the files and execute permission on the directory.
3. Add ACL entries that ensure any new files or directories in the **cases** directory have the correct permissions applied for *all* authorized users and groups.
4. Verify that you have made your ACL and file-system changes correctly.

   Use **ls** and **getfacl** to review your settings on **/shares/cases**.

   As the `student` user, use **su - *user*** to switch first to `manager1` and then to `contractor1`. Verify that you can write to a file, read from a file, make a directory, and write to a file in the new directory. Use **ls** to check the new directory permissions and **getfacl** to review the new directory ACL.

   As the `student` user, use **su - contractor3** to switch user. Try writing to a file (it should fail) and try to make a new directory (it should fail). As the `contractor3` user, you should be able to read from the **shortlist.txt** file in the **cases** directory and you should be able to read from the "test" files written in either of the new directories created by `manager1` and `contractor1` users.

> **NOTE**
> The set of tests above are some of the tests you could perform to check that access permissions are correct. You should devise appropriate access validation tests for your environment.

Evaluation

On `workstation`, run the **lab acl-review grade** command to confirm success of this exercise.

```
[student@workstation ~]$ lab acl-review grade
```

Finish

On `workstation`, run the **`lab acl-review finish`** command to complete this exercise.

```
[student@workstation ~]$ lab acl-review finish
```

This concludes the lab.

SOLUTION

CONTROLLING ACCESS TO FILES WITH ACLS

PERFORMANCE CHECKLIST

In this lab, you will set up a collaborative directory for users in two groups, combining the set-GID permission and default ACL entries to provide correct access permissions.

OUTCOMES

You should be able to:

- Configure set-GID permission on a folder, to inherit group ownership on files and folders inside.
- Configure ACL entries to allow or deny read/write/execute permissions to users and groups on files and directories.
- Configure default ACL to get the right ACL and file permissions automatically, on new files and directories.

BEFORE YOU BEGIN

Log in to `workstation` as `student` using `student` as the password.

On `workstation`, run the **`lab acl-review start`** command. This command runs a start script that determines if the `serverb` machine is reachable on the network. It also creates the users, groups, directories and files used in this exercise.

```
[student@workstation ~]$ lab acl-review start
```

A stock finance agency is setting up a collaborative share directory to hold case files, which members of the `managers` group will have read and write permissions on.

The co-founder of the agency, `manager1`, has decided that members of the `contractors` group should also be able to read and write to the share directory. However, `manager1` does not trust the `contractor3` user (a member of the `contractors` group), and as such, `contractor3` should have access to the directory restricted to read-only.

`manager1` has created the users and groups, and has started the process of setting up the share directory, copying in some templates files. Because `manager1` has been too busy, it falls to you to finish the job.

Your task is to complete the setup of the share directory. The directory and all of its contents should be owned by the `managers` group, with the files updated to read and write for the owner and group (`managers`). Other users should have no permissions. You also need to provide read and write permissions for the `contractors` group, with the exception of `contractor3`, who only gets read permissions. Make sure your setup applies to existing and future files.

Important information:

- Share directory: **/shares/cases** on `serverb`.
- The `manager1` and `manager2` users are members of the `managers` group.
- The `contractor1`, `contractor2`, and `contractor3` users are members of the `contractors` group.
- Two files exist in the directory: **shortlist.txt** and **backlog.txt**.
- All five user passwords are `redhat`.
- All changes should occur to the **/shares/cases** directory and its files; do not adjust the **/shares** directory.

1. The **cases** directory and its contents should belong to the `managers` group. New files added to the **cases** directory should automatically belong to the `managers` group. The user and group owners for the existing files should have read and write permission, and other users should have no permission at all.

> **NOTE**
> Hint: Do not use **setfacl**.

1.1. Log in to `serverb` as the `student` user.

```
[student@workstation ~]$ ssh student@serverb
...output omitted...
[student@serverb ~]$
```

1.2. Use the **sudo -i** command to switch to the `root` user. The password for the `student` user is `student`.

```
[student@serverb ~]$ sudo -i
[sudo] password for student: student
[root@serverb ~]#
```

1.3. Use the **chgrp** command to recursively update group ownership on the directory and its contents.

```
[root@serverb ~]# chgrp -R managers /shares/cases
```

1.4. Use the **chmod** command to update the **set-GID** flag on the directory.

```
[root@serverb ~]# chmod g+s /shares/cases
```

1.5. Use **chmod** to update all existing file permissions to **rw** for owner and group.

```
[root@serverb ~]# chmod 660 /shares/cases/*
```

2. Add ACL entries to the **cases** directory (and its contents) that allow members of the `contractors` group to have read/write access on the files and execute permission on the directory. Restrict the `contractor3` user to read access on the files and execute permission on the directory.

2.1. Use **setfacl** to recursively update the existing **cases** directory and its contents. Grant the contractors group read, write, and conditional execute permissions.

```
[root@serverb ~]# setfacl -Rm g:contractors:rwX /shares/cases
```

2.2. Use **setfacl** to recursively update the existing **cases** directory and its contents. Grant the contractor3 user read and conditional execute permissions.

```
[root@serverb ~]# setfacl -Rm u:contractor3:rX /shares/cases
```

3. Add ACL entries that ensure any new files or directories in the **cases** directory have the correct permissions applied for *all* authorized users and groups.

3.1. Use **setfacl** to update the *default* permissions for members of the contractors group. Default permissions are read, write, and execute (needed for proper subdirectory creation and access).

```
[root@serverb ~]# setfacl -m d:g:contractors:rwx /shares/cases
```

3.2. Use **setfacl** to update the *default* permissions for the contractor3 user. Default permissions are read and execute (needed for proper subdirectory access).

```
[root@serverb ~]# setfacl -m d:u:contractor3:rx /shares/cases
```

4. Verify that you have made your ACL and file-system changes correctly.

Use **ls** and **getfacl** to review your settings on **/shares/cases**.

As the student user, use **su - *user*** to switch first to manager1 and then to contractor1. Verify that you can write to a file, read from a file, make a directory, and write to a file in the new directory. Use **ls** to check the new directory permissions and **getfacl** to review the new directory ACL.

As the student user, use **su - contractor3** to switch user. Try writing to a file (it should fail) and try to make a new directory (it should fail). As the contractor3 user, you should be able to read from the **shortlist.txt** file in the **cases** directory and you should be able to read from the "test" files written in either of the new directories created by manager1 and contractor1 users.

4.1. As the root user, use **ls** to check the **cases** directory and its content. Look for group ownership, directory and file permissions. The "**s**" in the group file permissions

indicates the **set-GID** flag is set, and the "**+**" indicates that ACL entries exist. At the end exit from the `root` user session.

```
[root@serverb ~]# ls -ld /shares/cases
drwxrws---+ 2 root managers 46 Mar 29 00:40 /shares/cases
[root@serverb ~]# ls -l /shares/cases
total 8
-rw-rw----+ 1 root managers 44 Mar 29 00:33 backlog.txt
-rw-rw----+ 1 root managers 46 Mar 29 00:33 shortlist.txt
```

4.2. Use **getfacl** and review its output. Look for the named user and named group entries in both the standard and default ACL.

```
[root@serverb ~]# getfacl /shares/cases
# file: shares/cases
# owner: root
# group: managers
# flags: -s-
user::rwx
user:contractor3:r-x
group::rwx
group:contractors:rwx
mask::rwx
other::---
default:user::rwx
default:user:contractor3:r-x
default:group::rwx
default:group:contractors:rwx
default:mask::rwx
default:other::---

[root@serverb ~]# exit
logout
```

4.3. Switch to the `manager1` user and perform the following operations. Check that you get the expected access behavior.

```
[student@serverb ~]$ su - manager1
Password: redhat
[manager1@serverb ~]$ cd /shares/cases
[manager1@serverb cases]$ echo hello > manager1.txt
[manager1@serverb cases]$ cat shortlist.txt
###Shortlist of Clients to call###TEMPLATE###
[manager1@serverb cases]$ mkdir manager1.dir
[manager1@serverb cases]$ echo hello > manager1.dir/test.txt
[manager1@serverb cases]$ ls -ld manager1.dir
drwxrws---+ 2 manager1 managers 22 Mar 29 00:59 manager1.dir
[manager1@serverb cases]$ ls -l manager1.dir
total 4
-rw-rw----+ 1 manager1 managers 6 Mar 29 00:59 test.txt
[manager1@serverb cases]$ getfacl manager1.dir
# file: manager1.dir/
# owner: manager1
```

```
# group: managers
# flags: -s-
user::rwx
user:contractor3:r-x
group::rwx
group:contractors:rwx
mask::rwx
other::---
default:user::rwx
default:user:contractor3:r-x
default:group::rwx
default:group:contractors:rwx
default:mask::rwx
default:other::---

[manager1@serverb cases]$ exit
logout
```

4.4. Switch to the `contractor1` user and perform the following operations. Check that you get the expected access behavior.

```
[student@serverb ~]$ su - contractor1
Password: redhat
[contractor1@serverb ~]$ cd /shares/cases
[contractor1@serverb cases]$ echo hello > manager1.txt
[contractor1@serverb cases]$ cat shortlist.txt
###Shortlist of Clients to call###TEMPLATE###
[contractor1@serverb cases]$ mkdir contractor1.dir
[contractor1@serverb cases]$ echo hello > contractor1.dir/test.txt
[contractor1@serverb cases]$ ls -ld contractor1.dir
drwxrws---+ 2 contractor1 managers 22 Mar 29 01:05 contractor1.dir
[contractor1@serverb cases]$ ls -l contractor1.dir
total 4
-rw-rw----+ 1 contractor1 managers 6 Mar 29 01:07 test.txt
[manager1@serverb cases]$ getfacl contractor1.dir
# file: contractor1.dir/
# owner: contractor1
# group: managers
# flags: -s-
user::rwx
user:contractor3:r-x
group::rwx
group:contractors:rwx
mask::rwx
other::---
default:user::rwx
default:user:contractor3:r-x
default:group::rwx
default:group:contractors:rwx
default:mask::rwx
```

```
default:other::---

[contractor1@serverb cases]$ exit
logout
```

4.5. Switch to the `contractor3` user, and perform the following operations. Check that you get the expected access behavior.

```
[student@serverb ~]# su - contractor3
Password: redhat
[contractor3@serverb ~]# cd /shares/cases
[contractor3@serverb cases]# echo hello > contractor3.txt
-bash: contractor3.txt: Permission denied
[contractor3@serverb cases]# cat shortlist.txt
###Shortlist of Clients to call###TEMPLATE###
[contractor3@serverb cases]# mkdir contractor3.dir
mkdir: cannot create directory 'contractor3.dir': Permission denied
[contractor3@serverb cases]# cat manager1.dir/test.txt
hello
[contractor3@serverb cases]# cat contractor1.dir/test.txt
hello
[contractor3@serverb cases]# exit
logout
```

4.6. Log off from `serverb`

```
[student@serverb ~]# exit
logout
Connection to serverb closed.
[student@workstation ~]$
```

NOTE
The set of tests above are some of the tests you could perform to check that access permissions are correct. You should devise appropriate access validation tests for your environment.

Evaluation

On `workstation`, run the **`lab acl-review grade`** command to confirm success of this exercise.

```
[student@workstation ~]$ lab acl-review grade
```

Finish

On `workstation`, run the **`lab acl-review finish`** command to complete this exercise.

```
[student@workstation ~]$ lab acl-review finish
```

This concludes the lab.

SUMMARY

In this chapter, you learned:

- ACLs provide fine-grained access control to files and directories.
- The **`getfacl`** command displays the ACLs on a file or directory.
- The **`setfacl`** command sets, modifies, and removes default and standard ACLs on files and directories.
- Use default ACLs for controlling new files and directories permissions.
- Red Hat Enterprise Linux uses `systemd` and `udev` to apply predefined ACLs on devices, folders, and files.

CHAPTER 5

MANAGING SELINUX SECURITY

GOAL

Protect and manage the security of a server by using SELinux.

OBJECTIVES

- Describe how SELinux protects resources and how to select the enforcement mode.
- Configure a file's SELinux context to control how processes interact with that file.
- Configure SELinux booleans to allow runtime policy changes for varying access needs.
- Investigate SELinux log messages and troubleshoot SELinux AVC denials.

SECTIONS

- Changing the SELinux Enforcement Mode (and Guided Exercise)
- Controlling SELinux File Contexts (and Guided Exercise)
- Adjusting SELinux Policy with Booleans (and Guided Exercise)
- Investigating and Resolving SELinux Issues (and Guided Exercise)

LAB

Managing SELinux Security

CHANGING THE SELINUX ENFORCEMENT MODE

OBJECTIVES

After completing this section, you should be able to:

- Explain how SELinux protects resources.
- Change the current SELinux mode of a system.
- Set the default SELinux mode of a system.

HOW SELINUX PROTECTS RESOURCES

SELinux provides a critical security purpose in Linux, permitting or denying access to files and other resources that are significantly more precise than user permissions.

File permissions control which users or groups of users can access which specific files. However, a user given read or write access to any specific file can use that file in any way that user chooses, even if that use is not how the file should be used.

For example, with write access to a file, should a structured data file designed to be written to using only a particular program, be allowed to be opened and modified by other editors that could result in corruption?

File permissions cannot stop such undesired access. They were never designed to control *how* a file is used, but only *who* is allowed to read, write, or run a file.

SELinux consists of sets of policies, defined by the application developers, that declare exactly what actions and accesses are proper and allowed for each binary executable, configuration file, and data file used by an application. This is known as a *targeted policy* because one policy is written to cover the activities of a single application. Policies declare predefined labels that are placed on individual programs, files, and network ports.

WHY USE SECURITY ENHANCED LINUX?

Not all security issues can be predicted in advance. SELinux enforces a set of access rules preventing a weakness in one application from affecting other applications or the underlying system. SELinux provides an extra layer of security; it also adds a layer of complexity which can be off-putting to people new to this subsystem. Learning to work with SELinux may take time but the enforcement policy means that a weakness in one part of the system does not spread to other parts. If SELinux works poorly with a particular subsystem, you can turn off enforcement for that specific service until you find a solution to the underlying problem.

SELinux has three modes:

- Enforcing: SELinux is enforcing access control rules. Computers generally run in this mode.
- Permissive: SELinux is active but instead of enforcing access control rules, it records warnings of rules that have been violated. This mode is used primarily for testing and troubleshooting.
- Disabled: SELinux is turned off entirely: no SELinux violations are denied, nor even recorded. Discouraged!

BASIC SELINUX SECURITY CONCEPTS

Security Enhanced Linux (SELinux) is an additional layer of system security. The primary goal of SELinux is to protect user data from system services that have been compromised. Most Linux administrators are familiar with the standard user/group/other permission security model. This is a user and group based model known as discretionary access control. SELinux provides an additional layer of security that is object-based and controlled by more sophisticated rules, known as mandatory access control.

To allow remote anonymous access to a web server, firewall ports must be opened. However, this gives malicious people an opportunity to crack the system through a security exploit. If they succeed in compromising the web server process they gain its permissions. Specifically, the permissions of the `apache` user and the `apache` group. That user and group has read access to the document root, **`/var/www/html`**. It also has access to **`/tmp`**, and **`/var/tmp`**, and any other files and directories that are world writable.

SELinux is a set of security rules that determine which process can access which files, directories, and ports. Every file, process, directory, and port has a special security label called an SELinux *context*. A context is a name used by the SELinux policy to determine whether a process can access a file, directory, or port. By default, the policy does not allow any interaction unless an explicit rule grants access. If there is no allow rule, no access is allowed.

SELinux labels have several contexts: `user`, `role`, `type`, and `sensitivity`. The targeted policy, which is the default policy enabled in Red Hat Enterprise Linux, bases its rules on the third context: the type context. Type context names usually end with **`_t`**.

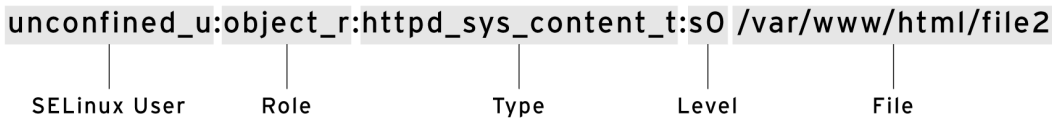

Figure 5.1: SELinux File Context

The type context for a web server is **`httpd_t`**. The type context for files and directories normally found in **`/var/www/html`** is **`httpd_sys_content_t`**. The contexts for files and directories normally found in **`/tmp`** and **`/var/tmp`** is **`tmp_t`**. The type context for web server ports is **`http_port_t`**.

Apache has a type context of **`httpd_t`**. There is a policy rule that permits Apache access to files and directories with the **`httpd_sys_content_t`** type context. By default files found in **`/var/www/html`** and other web server directories have the **`httpd_sys_content_t`** type context. There is no **`allow`** rule in the policy for files normally found in **`/tmp`** and **`/var/tmp`**, so access is not permitted. With SELinux enabled, a malicious user who had compromised the web server process could not access the **`/tmp`** directory.

The `MariaDB` server has a type context of **`mysqld_t`**. By default, files found in **`/data/mysql`** have the **`mysqld_db_t`** type context. This type context allows `MariaDB` access to those files but disables access by other services, such as the Apache web service.

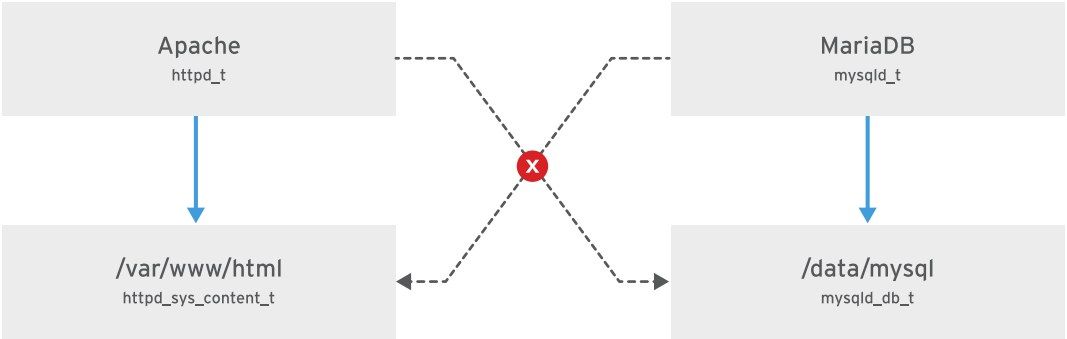

Figure 5.2: SELinux access

Many commands that deal with files use the **-Z** option to display or set SELinux contexts. For instance, **ps**, **ls**, **cp**, and **mkdir** all use the **-Z** option to display or set SELinux contexts.

```
[root@host ~]# ps axZ
LABEL                              PID TTY      STAT   TIME COMMAND
system_u:system_r:init_t:s0          1 ?        Ss     0:09 /usr/lib/systemd/...
system_u:system_r:kernel_t:s0        2 ?        S      0:00 [kthreadd]
system_u:system_r:kernel_t:s0        3 ?        S      0:00 [ksoftirqd/0]
...output omitted...
[root@host ~]# systemctl start httpd
[root@host ~]# ps -ZC httpd
LABEL                              PID TTY          TIME CMD
system_u:system_r:httpd_t:s0      1608 ?        00:00:05 httpd
system_u:system_r:httpd_t:s0      1609 ?        00:00:00 httpd
...output omitted...
[root@host ~]# ls -Z /home
drwx------. root    root    system_u:object_r:lost_found_t:s0 lost+found
drwx------. student student unconfined_u:object_r:user_home_dir_t:s0 student
drwx------. visitor visitor unconfined_u:object_r:user_home_dir_t:s0 visitor
[root@host ~]# ls -Z /var/www
drwxr-xr-x. root root system_u:object_r:httpd_sys_script_exec_t:s0 cgi-bin
drwxr-xr-x. root root system_u:object_r:httpd_sys_content_t:s0 error
drwxr-xr-x. root root system_u:object_r:httpd_sys_content_t:s0 html
drwxr-xr-x. root root system_u:object_r:httpd_sys_content_t:s0 icons
```

CHANGING THE CURRENT SELINUX MODE

The SELinux subsystem provides tools to display and change modes. To determine the current SELinux mode, run the **getenforce** command. To set SELinux to a different mode, use the **setenforce** command:

```
[user@host ~]# getenforce
Enforcing
[user@host ~]# setenforce
usage:  setenforce [ Enforcing | Permissive | 1 | 0 ]
[user@host ~]# setenforce 0
[user@host ~]# getenforce
Permissive
```

```
[user@host ~]# setenforce Enforcing
[user@host ~]# getenforce
Enforcing
```

Alternatively, you can set the SELinux mode at boot time by passing a parameter to the kernel: the kernel argument of **enforcing=0** boots the system into permissive mode; a value of **enforcing=1** sets enforcing mode. You can also disable SELinux completely by passing on the kernel parameter **selinux=0**. A value of **selinux=1** enables SELinux.

SETTING THE DEFAULT SELINUX MODE

You can also configure SELinux persistently using the **/etc/selinux/config** file. In the example below (the default configuration), the configuration file sets SELinux to **enforcing**. The comments also show the other valid values: **permissive** and **disabled**.

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX=enforcing
# SELINUXTYPE= can take one of these two values:
#     targeted - Targeted processes are protected,
#     minimum - Modification of targeted policy. Only selected processes
#               are protected.
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

The system reads this file at boot time and configures SELinux as shown. Kernel arguments (**selinux=0|1** and **enforcing=0|1**) override this configuration.

REFERENCES

getenforce(8), **setenforce**(8), and **selinux_config**(5) man pages

GUIDED EXERCISE

CHANGING THE SELINUX ENFORCEMENT MODE

In this lab, you will manage SELinux modes, both temporarily and persistently.

OUTCOMES

You should be able to view and set the current SELinux mode.

BEFORE YOU BEGIN

Log in as the `student` user on `workstation` using `student` as the password.

On `workstation`, run the **`lab selinux-opsmode start`** command. This command runs a start script that determines if the `servera` machine is reachable on the network.

```
[student@workstation ~]$ lab selinux-opsmode start
```

1. Use the **`ssh`** command to log in to `servera` as the `student` user. The systems are configured to use SSH keys for authentication, so a password is not required.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

2. Use the **`sudo -i`** command to switch to the `root` user. The password for the `student` user is `student`.

```
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

3. Change the default SELinux mode to permissive and reboot.

   3.1. Use the **`getenforce`** command to verify that `servera` is in enforcing mode.

```
[root@servera ~]# getenforce
Enforcing
```

   3.2. Use the **`vim`** command to open the **`/etc/selinux/config`** configuration file. Change the `SELINUX` parameter from `enforcing` to `permissive`.

```
[root@servera ~]# vim /etc/selinux/config
```

3.3. Use the **grep** command to confirm that the SELINUX parameter is set to permissive.

```
[root@servera ~]# grep '^SELINUX' /etc/selinux/config
SELINUX=permissive
SELINUXTYPE=targeted
```

3.4. Use the **systemctl reboot** command to reboot servera.

```
[root@servera ~]# systemctl reboot
Connection to servera closed by remote host.
Connection to servera closed.
[student@workstation ~]$
```

4. servera takes a few minutes to reboot. After a few minutes, log in to servera as the student user. Use the **sudo -i** command to become root. Display the current SELinux mode using the **getenforce** command.

4.1. From workstation using the **ssh** command log in to servera as the student user.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

4.2. Use the **sudo -i** command to become root.

```
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

4.3. Display the current SELinux mode using the **getenforce** command.

```
[root@servera ~]# getenforce
Permissive
```

5. In the **/etc/selinux/config** file, change the default SELinux mode to enforcing. This change only takes effect on next reboot.

   5.1. Use the **vim** command to open the **/etc/selinux/config** configuration file. Change the SELINUX back to enforcing.

```
[root@servera ~]# vim /etc/selinux/config
```

   5.2. Use the **grep** command to confirm that the SELINUX parameter is set to enforcing.

```
[root@servera ~]# grep '^SELINUX' /etc/selinux/config
SELINUX=enforcing
SELINUXTYPE=targeted
```

6. Use the **setenforce** command to set the current SELinux mode to enforcing without rebooting. Confirm that the mode has been set to enforcing using the **getenforce** command.

```
[root@servera ~]# setenforce 1
[root@servera ~]# getenforce
Enforcing
```

7. Exit from servera.

```
[root@servera ~]# exit
logout
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```

Finish

On workstation, run the **lab selinux-opsmode finish** script to complete this exercise.

```
[student@workstation ~]$ lab selinux-opsmode finish
```

This concludes the guided exercise.

CONTROLLING SELINUX FILE CONTEXTS

OBJECTIVES

After completing this section, you should be able to:

- Manage the SELinux policy rules that determine the default context for files and directories using the **semanage fcontext** command.
- Apply the context defined by the SELinux policy to files and directories using the **restorecon** command.

INITIAL SELINUX CONTEXT

On systems running SELinux, all processes and files are labeled. The label represents the security relevant information, known as the SELinux context.

New files typically inherit their SELinux context from the parent directory, thus ensuring that they have the proper context.

But this inheritance procedure can be undermined in two different ways. First, if you create a file in a different location from the ultimate intended location and then move the file, the file still has the SELinux context of the directory where it was created, not the destination directory. Second, if you copy a file preserving the SELinux context, as with the **cp -a** command, the SELinux context reflects the location of the original file.

The following example demonstrates inheritance and its pitfalls. Consider these two files created in **/tmp**, one moved to **/var/www/html** and the second one copied to the same directory. Note the SELinux contexts on the files. The file that was moved to the **/var/www/html** directory retains the file context for the **/tmp** directory. The file that was copied to the **/var/www/html** directory inherited SELinux context from the **/var/www/html** directory.

The **ls -Z** command displays the SELinux context of a file. Note the label of the file.

```
[root@host ~]# ls -Z /var/www/html/index.html
-rw-r--r--. root root unconfined_u:object_r:httpd_sys_content_t:s0 /var/www/html/
index.html
```

And the **ls -Zd** command displays the SELinux context of a directory:

```
[root@host ~]# ls -Zd /var/www/html/
drwxr-xr-x. root root system_u:object_r:httpd_sys_content_t:s0 /var/www/html/
```

Note that the **/var/www/html/index.html** has the same label as the parent directory **/var/www/html/**. Now, create files outside of the **/var/www/html** directory and note their file context:

```
[root@host ~]# touch /tmp/file1 /tmp/file2
[root@host ~]# ls -Z /tmp/file*
unconfined_u:object_r:user_tmp_t:s0 /tmp/file1
unconfined_u:object_r:user_tmp_t:s0 /tmp/file2
```

Move one of these files to the **/var/www/html** directory, copy another, and note the label of each:

```
[root@host ~]# mv /tmp/file1 /var/www/html/
[root@host ~]# cp /tmp/file2 /var/www/html/
```

```
[root@host ~]# ls -Z /var/www/html/file*
unconfined_u:object_r:user_tmp_t:s0 /var/www/html/file1
unconfined_u:object_r:httpd_sys_content_t:s0 /var/www/html/file2
```

The moved file maintains its original label while the copied file inherits the label from the **/var/www/html** directory. **unconfined_u:** is the user, **object_r:** denotes the role, and **s0** is the level. A sensitivity level of 0 is the lowest possible sensitivity level.

CHANGING THE SELINUX CONTEXT OF A FILE

Commands to change the SELinux context on files include **semanage fcontext**, **restorecon**, and **chcon**.

The preferred method to set the SELinux context for a file is to declare the default labeling for a file using the **semanage fcontext** command and then applying that context to the file using the **restorecon** command. This ensures that the labeling will be as desired even after a complete relabeling of the file system.

The **chcon** command changes SELinux contexts. **chcon** sets the security context on the file, stored in the file system. It is useful for testing and experimenting. However, it does not save context changes in the SELinux context database. When a **restorecon** command runs, changes made by the **chcon** command also do not survive. Also, if the entire file system is relabeled, the SELinux context for files changed using **chcon** are reverted.

The following screen shows a directory being created. The directory has a type value of **default_t**.

```
[root@host ~]# mkdir /virtual
[root@host ~]# ls -Zd /virtual
drwxr-xr-x. root root unconfined_u:object_r:default_t:s0 /virtual
```

The **chcon** command changes the file context of the **/virtual** directory: the type value changes to **httpd_sys_content_t**.

```
[root@host ~]# chcon -t httpd_sys_content_t /virtual
[root@host ~]# ls -Zd /virtual
drwxr-xr-x. root root unconfined_u:object_r:httpd_sys_content_t:s0 /virtual
```

The **restorecon** command runs and the type value returns to the value of **default_t**. Note the **Relabeled** message.

```
[root@host ~]# restorecon -v /virtual
Relabeled /virtual from unconfined_u:object_r:httpd_sys_content_t:s0 to
 unconfined_u:object_r:default_t:s0
[root@host ~]# ls -Zd /virtual
drwxr-xr-x. root root unconfined_u:object_r:default_t:s0 /virtual
```

DEFINING SELINUX DEFAULT FILE CONTEXT RULES

The **semanage fcontext** command displays and modifies the rules that **restorecon** uses to set default file contexts. It uses extended regular expressions to specify the path and file names. The most common extended regular expression used in **fcontext** rules is **(/.*)?**, which means "optionally, match a / followed by any number of characters". It matches the directory listed before the expression and everything in that directory recursively.

Basic File Context Operations

The following table is a reference for **semanage fcontext** options to add, remove or list SELinux file contexts.

semanage fcontext commands

OPTION	DESCRIPTION
-a, --add	Add a record of the specified object type
-d, --delete	Delete a record of the specified object type
-l, --list	List records of the specified object type

To ensure that you have the tools to manage SELinux contexts, install the **policycoreutil** package and the **policycoreutil-python** package if needed. These contain the **restorecon** command and **semanage** command, respectively.

To ensure that all files in a directory have the correct file context run the **semanage fcontext -l** followed by the **restorecon** command. In the following example, note the file context of each file before and after the **semanage** and **restorecon** commands run.

```
[root@host ~]# ls -Z /var/www/html/file*
unconfined_u:object_r:user_tmp_t:s0 /var/www/html/file1
 unconfined_u:object_r:httpd_sys_content_t:s0 /var/www/html/file2
```

```
[root@host ~]# semanage fcontext -l
...output omitted...
/var/www(/.*)?       all files    system_u:object_r:httpd_sys_content_t:s0
...output omitted...
```

```
[root@host; ~]# restorecon -Rv /var/www/
Relabeled /var/www/html/file1 from unconfined_u:object_r:user_tmp_t:s0 to
 unconfined_u:object_r:httpd_sys_content_t:s0
[root@host ~]# ls -Z /var/www/html/file*
unconfined_u:object_r:httpd_sys_content_t:s0 /var/www/html/file1
 unconfined_u:object_r:httpd_sys_content_t:s0 /var/www/html/file2
```

The following example shows how to use **semanage** to add a context for a new directory.

```
[root@host ~]# mkdir /virtual
[root@host ~]# touch /virtual/index.html
[root@host ~]# ls -Zd /virtual/
drwxr-xr-x. root root unconfined_u:object_r:default_t:s0 /virtual/
```

```
[root@host ~]# ls -Z /virtual/
-rw-r--r--. root root unconfined_u:object_r:default_t:s0 index.html
[root@host ~]# semanage fcontext -a -t httpd_sys_content_t '/virtual(/.*)?'
[root@host ~]# restorecon -RFvv /virtual
[root@host ~]# ls -Zd /virtual/
drwxr-xr-x. root root system_u:object_r:httpd_sys_content_t:s0 /virtual/
[root@host ~]# ls -Z /virtual/
-rw-r--r--. root root system_u:object_r:httpd_sys_content_t:s0 index.html
```

REFERENCES

chcon(1), **restorecon**(8), **semanage**(8), and **semanage-fcontext**(8) man pages

GUIDED EXERCISE

CONTROLLING SELINUX FILE CONTEXTS

In this lab, you will make a persistent change to the SELinux context of a directory and its contents.

OUTCOMES

You should be able to configure the *Apache* HTTP server to publish web content from a non-standard document root.

BEFORE YOU BEGIN

Log in as the `student` user on `workstation` using `student` as the password.

On `workstation`, run the **`lab selinux-filecontexts start`** command. This command runs a start script that determines whether the `servera` machine is reachable on the network. It also installs the `httpd` service and configures the firewall on `servera` to allow HTTP connections.

```
[student@workstation ~]$ lab selinux-filecontexts start
```

1. Use the **`ssh`** command to log in to `servera` as the `student` user. The systems are configured to use SSH keys for authentication, so a password is not required.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

2. Use the **`sudo -i`** command to switch to the `root` user. The password for the `student` user is `student`.

```
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

3. Configure Apache to use a document root in a non-standard location.

   3.1. Create the new document root, **`/custom`** using the **`mkdir`** command.

```
[root@servera ~]# mkdir /custom
```

   3.2. Create the **`index.html`** file in the **`/custom`** document root using the **`echo`** command.

```
[root@servera ~]# echo 'This is SERVERA.' > /custom/index.html
```

3.3. Configure Apache to use the new document root location. You need to replace the two occurrences of **/var/www/html** with **/custom** in the Apache configuration file, **/etc/httpd/conf/httpd.conf**.

```
[root@servera ~]# vim /etc/httpd/conf/httpd.conf
[root@servera ~]# grep custom /etc/httpd/conf/httpd.conf
DocumentRoot "/custom"
<Directory "/custom">
```

4. Start and enable the Apache web service and confirm that the service is running.

4.1. Start and enable the Apache web service using the **systemctl** command.

```
[root@servera ~]# systemctl enable --now httpd
```

4.2. Use the **systemctl** command to confirm that the service is running.

```
[root@servera ~]# systemctl status httpd
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled;
 vendor preset: disabled)
   Active: active (running) since Mon 2019-03-25 19:16:48 CET; 15h ago
     Docs: man:httpd.service(8)
 Main PID: 6565 (httpd)
   Status: "Total requests: 16; Idle/Busy workers 100/0;Requests/sec:
 0.000285; Bytes served/sec:   0 B/sec"
   Tasks: 213 (limit: 11406)
   Memory: 37.3M
   CGroup: /system.slice/httpd.service
           ├─6565 /usr/sbin/httpd -DFOREGROUND
           ├─6566 /usr/sbin/httpd -DFOREGROUND
           ├─6567 /usr/sbin/httpd -DFOREGROUND
           ├─6568 /usr/sbin/httpd -DFOREGROUND
           └─6569 /usr/sbin/httpd -DFOREGROUND

Mar 25 19:16:48 servera.lab.example.com systemd[1]: Starting The
 Apache HTTP Server...
Mar 25 19:16:48 servera.lab.example.com httpd[6565]: Server
 configured, listening on: port 80
```

5. Open a web browser on `workstation` and try to view `http://servera/index.html`. You will get an error message that says you do not have permission to access the file.

6. To permit access to the **index.html** file on servera, SELinux must be configured. Define an SELinux file context rule that sets the context type to **httpd_sys_content_t** for the **/custom** directory and all the files below it.

```
[root@servera ~]# semanage fcontext -a -t httpd_sys_content_t '/
custom(/.*)?'
```

7. Use the **restorecon** command to change the file contexts.

```
[root@servera ~]# restorecon -Rv /custom
Relabeled /custom from unconfined_u:object_r:default_t:s0 to
 unconfined_u:object_r:httpd_sys_content_t:s0
Relabeled /custom/index.html from unconfined_u:object_r:default_t:s0 to
 unconfined_u:object_r:httpd_sys_content_t:s0
```

8. Try to view http://servera/index.html again. You should see the message **This is SERVERA.** displayed.

9. Exit from servera.

```
[root@servera ~]# exit
logout
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```

Finish

On workstation, run the **lab selinux-filecontexts finish** script to complete this exercise.

```
[student@workstation ~]$ lab selinux-filecontexts finish
```

This concludes the guided exercise.

ADJUSTING SELINUX POLICY WITH BOOLEANS

OBJECTIVES

After completing this section, you should be able to:

- Activate and deactivate SELinux policy rules using **`setsebool`**.
- Manage the persistent value of SELinux booleans using the **`semanage boolean -l`** command.
- Consult man pages that end with **`_selinux`** to find useful information about SELinux booleans.

SELINUX BOOLEANS

SELinux booleans are switches that change the behavior of the SELinux policy. SELinux booleans are rules that can be enabled or disabled. They can be used by security administrators to tune the policy to make selective adjustments.

The SELinux man pages, provided with the *selinux-policy-doc* package, describe the purpose of the available booleans. The **`man -k '_selinux'`** command lists these man pages.

Commands useful for managing SELinux booleans include **`getsebool`**, which lists booleans and their state, and **`setsebool`** which modifies booleans. **`setsebool -P`** modifies the SELinux policy to make the modification persistent. And **`semanage boolean -l`** reports on whether or not a boolean is persistent, along with a short description of the boolean.

Non-privileged users can run the **`getsebool`** command, but you must be a superuser to run **`semanage boolean -l`** and **`setsebool -P`**.

```
[user@host ~]$ getsebool -a
abrt_anon_write --> off
abrt_handle_event --> off
abrt_upload_watch_anon_write --> on
antivirus_can_scan_system --> off
antivirus_use_jit --> off
...output omitted...
[user@host ~]$ getsebool httpd_enable_homedirs
httpd_enable_homedirs --> off
```

```
[user@host ~]$ setsebool httpd_enable_homedirs on
Could not change active booleans. Please try as root: Permission denied
[user@host ~]$ sudo setsebool httpd_enable_homedirs on
[user@host ~]$ sudo semanage boolean -l | grep httpd_enable_homedirs
httpd_enable_homedirs          (on   ,  off)  Allow httpd to enable homedirs
[user@host ~]$ getsebool httpd_enable_homedirs
httpd_enable_homedirs --> on
```

The **`-P`** option writes all pending values to the policy, making them persistent across reboots. In the example that follows, note the values in parentheses: both are now set to **`on`**.

```
[user@host ~]$ setsebool -P httpd_enable_homedirs on
[user@host ~]$ sudo semanage boolean -l | grep httpd_enable_homedirs
httpd_enable_homedirs          (on   ,   on)  Allow httpd to enable homedirs
```

To list booleans in which the current state differs from the default state, run **`semanage boolean -l -C`**.

```
[user@host ~]$ sudo semanage boolean -l -C
SELinux boolean                State  Default Description

cron_can_relabel               (off   ,   on)  Allow cron to can relabel
```

REFERENCES

`booleans`(8), **`getsebool`**(8), **`setsebool`**(8), **`semanage`**(8), **`semanage-boolean`**(8) man pages

GUIDED EXERCISE

ADJUSTING SELINUX POLICY WITH BOOLEANS

Apache can publish web content hosted in users' home directories, but SELinux prevents this by default. In this exercise, you will identify and change the SELinux boolean that permits Apache to access user home directories.

OUTCOMES

You should be able to configure *Apache* to publish web content from users' home directories.

BEFORE YOU BEGIN

Log in as the `student` user on `workstation` using `student` as the password.

On `workstation`, run the **`lab selinux-booleans start`** command. This command runs a start script that determines whether the `servera` machine is reachable on the network. It also installs the `httpd` service and configures the firewall on `servera` to allow HTTP connections.

```
[student@workstation ~]$ lab selinux-booleans start
```

1. Use the **`ssh`** command to log in to `servera` as the `student` user. The systems are configured to use SSH keys for authentication, so a password is not required.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

2. Use the **`sudo -i`** command to switch to the `root` user. The password for the `student` user is `student`.

```
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

3. To enable the Apache feature that permits users to publish web content from their home directories, you must edit the **`/etc/httpd/conf.d/userdir.conf`** configuration file. Comment out the line that sets **`UserDir`** to **`disabled`** and uncomment the line that sets **`UserDir`** to **`public_html`**.

```
[root@servera ~]# vim /etc/httpd/conf.d/userdir.conf
#UserDir disabled
UserDir public_html
```

4. Use the **grep** command to confirm the changes.

```
[root@servera ~]# grep '#UserDir' /etc/httpd/conf.d/userdir.conf
#UserDir disabled
[root@servera ~]# grep '^ *UserDir' /etc/httpd/conf.d/userdir.conf
UserDir public_html
```

5. Start and enable the Apache web service to make the changes take effect.

```
[root@servera ~]# systemctl enable --now httpd
```

6. In another terminal window log in as `student`. SSH into `servera`. Create some web content that is published from a user's home directory.

   6.1. In another terminal window log in as `student`. Use the **ssh** command to log in to `servera` as the `student` user.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

   6.2. Use the **mkdir** command to create a directory called **~/public_html**.

```
[student@servera ~]$ mkdir ~/public_html
```

   6.3. Create the **index.html** file with the following content:

```
[student@servera ~]$ echo 'This is student content on SERVERA.' > \
~/public_html/index.html
```

   6.4. Use the **chmod** command to change the permissions on `student`'s home directory so Apache can access the **public_html** subdirectory.

```
[student@servera ~]$ chmod 711 ~
```

7. Open a web browser on `workstation` and try to view the following URL: `http://servera/~student/index.html`. You get an error message that says you do not have permission to access the file.

8. In the terminal window with **root** access, use the **getsebool** command to see if there are any booleans that restrict access to home directories.

```
[root@servera ~]# getsebool -a | grep home
...output omitted...
httpd_enable_homedirs --> off
...output omitted...
```

9. In the terminal window with **root** access, use the **setsebool** command to enable home directory access persistently.

```
[root@servera ~]# setsebool -P httpd_enable_homedirs on
```

10. Try to view `http://servera/~student/index.html` again. You should see the message: **This is student content on SERVERA.**

11. Exit from `servera`.

```
[root@servera ~]# exit
logout
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```

Finish

On `workstation`, run the **lab selinux-booleans finish** script to complete this exercise.

```
[student@workstation ~]$ lab selinux-booleans finish
```

This concludes the guided exercise.

INVESTIGATING AND RESOLVING SELINUX ISSUES

OBJECTIVES

After completing this section, you should be able to:

- Use SELinux log analysis tools.
- Display useful information during SELinux troubleshooting using the **sealert** command.

TROUBLESHOOTING SELINUX ISSUES

It is important to understand what actions you must take when SELinux prevents access to files on a server that you know should be accessible. Use the following steps as a guide to troubleshooting these issues:

1. Before thinking of making any adjustments, consider that SELinux may be doing its job correctly by prohibiting the attempted access. If a web server tries to access files in **/home**, this could signal a compromise of the service if web content is not published by users. If access should have been granted, then additional steps need to be taken to solve the problem.
2. The most common SELinux issue is an incorrect file context. This can occur when a file is created in a location with one file context and moved into a place where a different context is expected. In most cases, running **restorecon** will correct the issue. Correcting issues in this way has a very narrow impact on the security of the rest of the system.
3. Another remedy for overly restrictive access could be the adjustment of a Boolean. For example, the **ftpd_anon_write** boolean controls whether anonymous FTP users can upload files. You must turn this boolean on to permit anonymous FTP users to upload files to a server. Adjusting booleans requires more care because they can have a broad impact on system security.
4. It is possible that the SELinux policy has a bug that prevents a legitimate access. Since SELinux has matured, this is a rare occurrence. When it is clear that a policy bug has been identified, contact Red Hat support to report the bug so it can be resolved.

MONITORING SELINUX VIOLATIONS

Install the *setroubleshoot-server* package to send SELinux messages to **/var/log/messages**. **setroubleshoot-server** listens for audit messages in **/var/log/audit/audit.log** and sends a short summary to **/var/log/messages**. This summary includes *unique identifiers (UUID)* for SELinux violations that can be used to gather further information. The **sealert -l *UUID*** command is used to produce a report for a specific incident. Use **sealert -a /var/log/audit/audit.log** to produce reports for all incidents in that file.

Consider the following sample sequence of commands on a standard Apache web server:

```
[root@host ~]# touch /root/file3
[root@host ~]# mv /root/file3 /var/www/html
[root@host ~]# systemctl start httpd
[root@host ~]# curl http://localhost/file3
```

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>403 Forbidden</title>
</head><body>
<h1>Forbidden</h1>
<p>You don't have permission to access /file3
on this server.</p>
</body></html>
```

You expect the web server to deliver the contents of **`file3`** but instead it returns a **`permission denied`** error. Inspecting both **`/var/log/audit/audit.log`** and **`/var/log/messages`** reveals extra information about this error.

```
[root@host ~]# tail /var/log/audit/audit.log
...output omitted...
type=AVC msg=audit(1392944135.482:429): avc:  denied  { getattr } for
  pid=1609 comm="httpd" path="/var/www/html/file3" dev="vda1" ino=8980981
  scontext=system_u:system_r:httpd_t:s0
  tcontext=unconfined_u:object_r:admin_home_t:s0 tclass=file
...output omitted...
[root@host ~]# tail /var/log/messages
...output omitted...
Feb 20 19:55:42 host setroubleshoot: SELinux is preventing /usr/sbin/httpd
  from getattr access on the file . For complete SELinux messages. run
  sealert -l 613ca624-248d-48a2-a7d9-d28f5bbe2763
```

Both log files indicate that an SELinux denial is the culprit. The **`sealert`** command that is part of the output in **`/var/log/messages`** provides extra information, including a possible fix.

```
[root@host ~]# sealert -l 613ca624-248d-48a2-a7d9-d28f5bbe2763
SELinux is preventing /usr/sbin/httpd from getattr access on the file .

*****  Plugin catchall (100. confidence) suggests   **************************

If you believe that httpd should be allowed getattr access on the
  file by default.
Then you should report this as a bug.
You can generate a local policy module to allow this access.
Do
allow this access for now by executing:
# grep httpd /var/log/audit/audit.log | audit2allow -M mypol
# semodule -i mypol.pp


Additional Information:
Source Context                system_u:system_r:httpd_t:s0
Target Context                unconfined_u:object_r:admin_home_t:s0
Target Objects                 [ file ]
Source                        httpd
Source Path                   /usr/sbin/httpd
Port                          <Unknown>
Host                          servera
Source RPM Packages           httpd-2.4.6-14.el7.x86_64
```

```
Target RPM Packages
Policy RPM                    selinux-policy-3.12.1-124.el7.noarch
Selinux Enabled               True
Policy Type                   targeted
Enforcing Mode                Enforcing
Host Name                     servera
Platform                      Linux servera 3.10.0-84.el7.x86_64 #1
                              SMP Tue Feb 4 16:28:19 EST 2014 x86_64 x86_64
Alert Count                   2
First Seen                    2014-02-20 19:55:35 EST
Last Seen                     2014-02-20 19:55:35 EST
Local ID                      613ca624-248d-48a2-a7d9-d28f5bbe2763

Raw Audit Messages
type=AVC msg=audit(1392944135.482:429): avc:  denied  { getattr } for
  pid=1609 comm="httpd" path="/var/www/html/file3" dev="vda1" ino=8980981
  scontext=system_u:system_r:httpd_t:s0
  tcontext=unconfined_u:object_r:admin_home_t:s0 tclass=file

type=SYSCALL msg=audit(1392944135.482:429): arch=x86_64 syscall=lstat
  success=no exit=EACCES a0=7f9fed0edea8 a1=7fff7bffc770 a2=7fff7bffc770
  a3=0 items=0 ppid=1608 pid=1609 auid=4294967295 uid=48 gid=48 euid=48
  suid=48 fsuid=48 egid=48 sgid=48 fsgid=48 tty=(none) ses=4294967295
  comm=httpd exe=/usr/sbin/httpd subj=system_u:system_r:httpd_t:s0 key=(null)

Hash: httpd,httpd_t,admin_home_t,file,getattr
```

NOTE

The **`Raw Audit Messages`** section reveals the target file that is the problem, **`/var/www/html/file3`**. Also, the target context, **`tcontext`**, does not look like it belongs with a web server. Use the **`restorecon /var/www/html/file3`** command to fix the file context. If there are other files that need to be adjusted, **`restorecon`** can recursively reset the context: **`restorecon -R /var/www/`**.

The **`Raw Audit Messages`** section of the **`sealert`** command contains information from **`/var/log/audit.log`**. To search the **`/var/log/audit.log`** file use the **`ausearch`** command. The **`-m`** searches on the message type. The **`-ts`** option searches based on time.

```
[root@host ~]# ausearch -m AVC -ts recent
----
time->Tue Apr  9 13:13:07 2019
type=PROCTITLE msg=audit(1554808387.778:4002):
 proctitle=2F7573722F7362696E2F6874747064002D44464F524547524F554E44
type=SYSCALL msg=audit(1554808387.778:4002): arch=c000003e syscall=49
 success=no exit=-13 a0=3 a1=55620b8c9280 a2=10 a3=7ffed967661c items=0
 ppid=1 pid=9340 auid=4294967295 uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0
 sgid=0 fsgid=0 tty=(none) ses=4294967295 comm="httpd" exe="/usr/sbin/httpd"
 subj=system_u:system_r:httpd_t:s0 key=(null)
type=AVC msg=audit(1554808387.778:4002): avc:  denied  { name_bind }
 for  pid=9340 comm="httpd" src=82 scontext=system_u:system_r:httpd_t:s0
 tcontext=system_u:object_r:reserved_port_t:s0 tclass=tcp_socket permissive=0
```

WEB CONSOLE

If *Web Console* is installed it can also be used for troubleshooting SELinux issues. Log on to Web Console and select ***SELinux*** from the menu on the left. The SELinux Policy window informs you of the current enforce policy. Any issues are detailed in the ***SELinux Access Control Errors*** section.

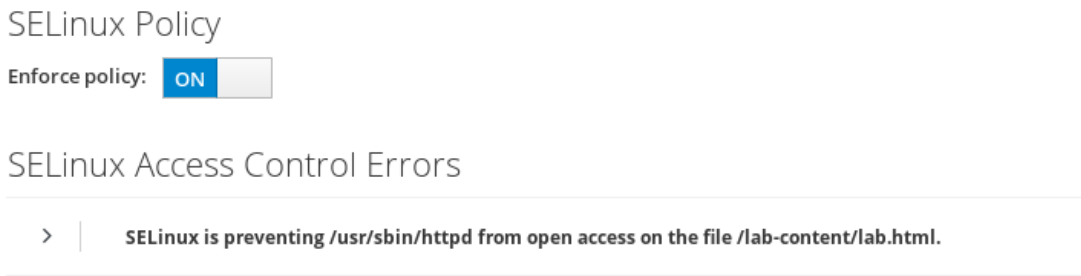

Figure 5.3: SELinux Policy in Web Console

Click on the **>** character to show error details. Click on ***solution details*** to show all details and possible solution.

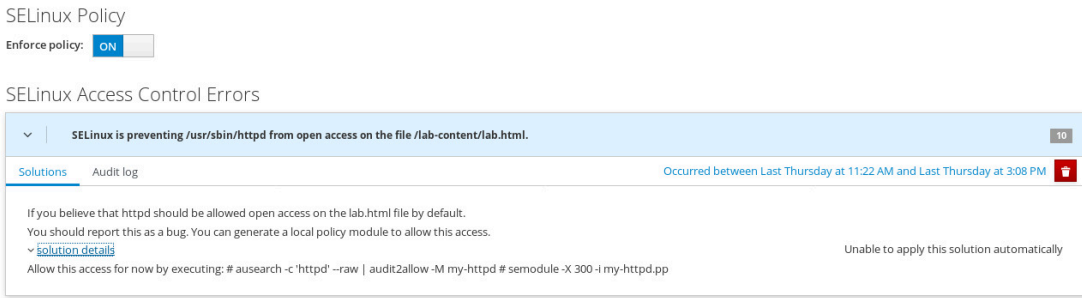

Figure 5.4: SELinux Policy Solution in Web Console

Once the problem has been solved, the ***SELinux Access Control Errors*** section should no longer show the error. If the message **`No SELinux alerts`** appears, then all issues have been fixed.

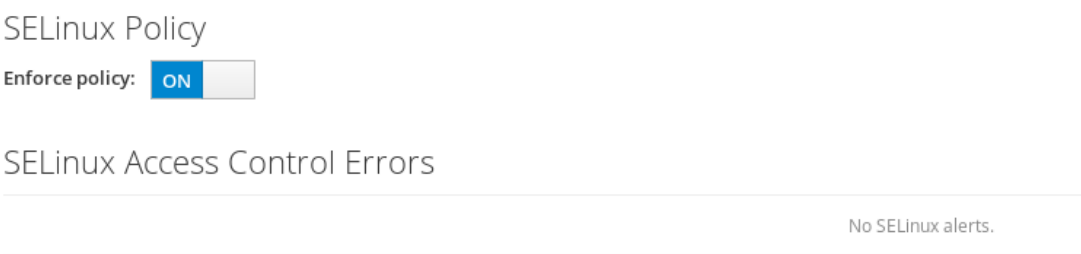

Figure 5.5: No SELinux Alerts in Web Console

REFERENCES

`sealert`(8) man page

GUIDED EXERCISE

INVESTIGATING AND RESOLVING SELINUX ISSUES

In this lab, you will learn how to troubleshoot SELinux security denials.

OUTCOMES

You will get some experience using SELinux troubleshooting tools.

BEFORE YOU BEGIN

Log in as the `student` user on `workstation` using `student` as the password.

On `workstation`, run the **`lab selinux-issues start`** command. This command runs a start script that determines whether the `servera` machine is reachable on the network. It installs the `httpd` service, configures the firewall on `servera` to allow HTTP connections, and removes the SELinux context for the **`/custom`** directory.

```
[student@workstation ~]$ lab selinux-issues start
```

1. Use the **`ssh`** command to log in to `servera` as the `student` user. The systems are configured to use SSH keys for authentication, so a password is not required.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

2. Use the **`sudo -i`** command to switch to the `root` user. The password for the `student` user is `student`.

```
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

3. Open a web browser on `workstation` and try to view `http://servera/index.html`. You will get an error message that says you do not have permission to access the file.

4. Using the **`less`** command, view the contents of **`/var/log/messages`**. Use the **`/`** key and search for **`sealert`**. Copy the suggested **`sealert`** command so that it can be used in the next step. Use the **`q`** key to quit the **`less`** command.

```
[root@servera ~]# less /var/log/messages
...output omitted...
Mar 28 06:07:03 servera setroubleshoot[15326]: SELinux is preventing /usr/
sbin/httpd from getattr access on the file /custom/index.html. For complete
 SELinux messages run: sealert -l b1c9cc8f-a953-4625-b79b-82c4f4f1fee3
Mar 28 06:07:03 servera platform-python[15326]: SELinux is
 preventing /usr/sbin/httpd from getattr access on the file /custom/
index.html.#012#012*****  Plugin catchall (100. confidence) suggests
 **************************#012#012If you believe that httpd should be
 allowed getattr access on the index.html file by default.#012Then you
 should report this as a bug.#012You can generate a local policy module to
 allow this access.#012Do#012allow this access for now by executing:#012#
 ausearch -c 'httpd' --raw | audit2allow -M my-httpd#012# semodule -X 300 -i
 my-httpd.pp#012
Mar 28 06:07:04 servera setroubleshoot[15326]: failed to retrieve rpm info
 for /custom/index.html
...output omitted...
```

5. Run the suggested **sealert** command. Note the source context, the target objects, the policy, and the enforcing mode.

```
[root@servera ~]# sealert -l b1c9cc8f-a953-4625-b79b-82c4f4f1fee3
SELinux is preventing /usr/sbin/httpd from getattr access on the file /
custom/index.html.

*****  Plugin catchall (100. confidence) suggests
 **************************

If you believe that httpd should be allowed getattr access on the index.html
 file by default.
Then you should report this as a bug.
You can generate a local policy module to allow this access.
Do
allow this access for now by executing:
# ausearch -c 'httpd' --raw | audit2allow -M my-httpd
# semodule -X 300 -i my-httpd.pp


Additional Information:
Source Context                system_u:system_r:httpd_t:s0
Target Context                unconfined_u:object_r:default_t:s0
Target Objects                /custom/index.html [ file ]
Source                        httpd
Source Path                   /usr/sbin/httpd
Port                          <Unknown>
Host                          servera.lab.example.com
Source RPM Packages
Target RPM Packages
Policy RPM                    selinux-policy-3.14.1-59.el8.noarch
Selinux Enabled               True
Policy Type                   targeted
Enforcing Mode                Enforcing
```

```
Host Name                     servera.lab.example.com
Platform                      Linux servera.lab.example.com
 4.18.0-67.el8.x86_64
                              #1 SMP Sat Feb 9 12:44:00 UTC 2019 x86_64
 x86_64
                              Alert Count                   18
                              First Seen                    2019-03-25
 19:25:28 CET
                              Last Seen                     2019-03-28
 11:07:00 CET
                              Local ID                      b1c9cc8f-
a953-4625-b79b-82c4f4f1fee3

                              Raw Audit Messages
                              type=AVC msg=audit(1553767620.970:16958):
 avc:  denied  { getattr } for  pid=15067 comm="httpd" path="/custom/
index.html" dev="vda1" ino=4208311 scontext=system_u:system_r:httpd_t:s0
 tcontext=unconfined_u:object_r:default_t:s0 tclass=file permissive=0


                              Hash: httpd,httpd_t,default_t,file,getattr
```

6. The **Raw Audit Messages** section of the **sealert** command contains information from the **/var/log/audit/audit.log**. Use the **ausearch** command to search the **/var/log/audit/audit.log** file. The **-m** option searches on the message type. The **-ts** option searches based on time. This entry identifies the relevant process and file causing the alert. The process is the **httpd** Apache web server, the file is **/custom/index.html**, and the context is **system_r:httpd_t**.

```
[root@servera ~]# ausearch -m AVC -ts recent
----
time->Thu Mar 28 13:39:30 2019
type=PROCTITLE msg=audit(1553776770.651:17000):
 proctitle=2F7573722F7362696E2F6874747064002D44464F524547524F554E44
type=SYSCALL msg=audit(1553776770.651:17000): arch=c000003e syscall=257
 success=no exit=-13 a0=ffffff9c a1=7f8db803f598 a2=80000 a3=0 items=0
 ppid=15063 pid=15065 auid=4294967295 uid=48 gid=48 euid=48 suid=48 fsuid=48
 egid=48 sgid=48 fsgid=48 tty=(none) ses=4294967295 comm="httpd" exe="/usr/
sbin/httpd" subj=system_u:system_r:httpd_t:s0 key=(null)
type=AVC msg=audit(1553776770.651:17000): avc:  denied
 { open } for  pid=15065 comm="httpd" path="/custom/index.html"
 dev="vda1" ino=4208311 scontext=system_u:system_r:httpd_t:s0
 tcontext=unconfined_u:object_r:default_t:s0 tclass=file permissive=0
```

7. To resolve the issue use the **semanage** and **restorecon** commands. The context to manage is **httpd_sys_content_t**.

```
[root@servera ~]# semanage fcontext -a -t httpd_sys_content_t '/
custom(/.*)?'
[root@servera ~]# restorecon -Rv /custom
Relabeled /custom from unconfined_u:object_r:default_t:s0 to
 unconfined_u:object_r:httpd_sys_content_t:s0
Relabeled /custom/index.html from unconfined_u:object_r:default_t:s0 to
 unconfined_u:object_r:httpd_sys_content_t:s0
```

8. Try to view `http://servera/index.html` again. You should see the message **This is SERVERA.** displayed.

9. Exit from `servera`.

```
[root@servera ~]# exit
logout
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```

Finish

On `workstation`, run the **lab selinux-issues finish** script to complete this exercise.

```
[student@workstation ~]$ lab selinux-issues finish
```

This concludes the guided exercise.

LAB

MANAGING SELINUX SECURITY

PERFORMANCE CHECKLIST

In this lab, you will solve an SELinux access denial problem. System administrators are having trouble getting a new web server to deliver content to clients when SELinux is in enforcing mode.

OUTCOMES

You should be able to:

- Identify issues in system log files.
- Adjust the SELinux configuration.

BEFORE YOU BEGIN

Log in to `workstation` as `student` using `student` as the password.

On `workstation`, run the **`lab selinux-review start`** command. This command runs a start script that determines whether the `serverb` machine is reachable on the network. It also installs the httpd *Apache* server, creates a new DocumentRoot for *Apache*, and updates the configuration file.

```
[student@workstation ~]$ lab selinux-review start
```

1. Log in to `serverb` as the root user.
2. Launch a web browser on `workstation` and browse to `http://serverb/lab.html`. You will see the error message: **`You do not have permission to access /lab.html on this server.`**
3. Research and identify the SELinux issue that is preventing Apache from serving web content.
4. Display the SELinux context of the new HTTP document root and the original HTTP document root. Resolve the SELinux issue preventing Apache from serving web content.
5. Verify that the SELinux issue has been resolved and Apache is able to serve web content.
6. Exit from `serverb`.

Evaluation

On `workstation`, run the **`lab selinux-review grade`** script to confirm success on this exercise.

```
[student@workstation ~]$ lab selinux-review grade
```

Finish

On `workstation`, run the **`lab selinux-review finish`** script to complete the lab.

```
[student@workstation ~]$ lab selinux-review finish
```

This concludes the lab.

SOLUTION

MANAGING SELINUX SECURITY

PERFORMANCE CHECKLIST

In this lab, you will solve an SELinux access denial problem. System administrators are having trouble getting a new web server to deliver content to clients when SELinux is in enforcing mode.

OUTCOMES

You should be able to:

- Identify issues in system log files.
- Adjust the SELinux configuration.

BEFORE YOU BEGIN

Log in to `workstation` as `student` using `student` as the password.

On `workstation`, run the **`lab selinux-review start`** command. This command runs a start script that determines whether the `serverb` machine is reachable on the network. It also installs the httpd *Apache* server, creates a new DocumentRoot for *Apache*, and updates the configuration file.

```
[student@workstation ~]$ lab selinux-review start
```

1. Log in to `serverb` as the root user.

   1.1. Use the **`ssh`** command to log in to `serverb` as the `student` user. The systems are configured to use SSH keys for authentication, so a password is not required.

   ```
   [student@workstation ~]$ ssh student@serverb
   ...output omitted...
   [student@serverb ~]$
   ```

   1.2. Use the **`sudo -i`** command to switch to the `root` user. The password for the `student` user is `student`.

   ```
   [student@serverb ~]$ sudo -i
   [sudo] password for student: student
   [root@serverb ~]#
   ```

2. Launch a web browser on `workstation` and browse to `http://serverb/lab.html`. You will see the error message: **`You do not have permission to access /lab.html on this server.`**

3. Research and identify the SELinux issue that is preventing Apache from serving web content.

3.1. Using the **less** command, view the contents of **/var/log/messages**. Use the **/** key and search for **sealert**. Use the **q** key to quit the **less** command.

```
[root@serverb ~]# less /var/log/messages
Mar 28 10:19:51 serverb setroubleshoot[27387]: SELinux is
 preventing /usr/sbin/httpd from getattr access on the file /lab-
content/lab.html. For complete SELinux messages run: sealert -l
 8824e73d-3ab0-4caf-8258-86e8792fee2d
Mar 28 10:19:51 serverb platform-python[27387]: SELinux is preventing /
usr/sbin/httpd from getattr access on the file /lab-content/
lab.html.#012#012*****  Plugin catchall (100. confidence) suggests
  **************************#012#012If you believe that httpd should
 be allowed getattr access on the lab.html file by default.#012Then
 you should report this as a bug.#012You can generate a local policy
 module to allow this access.#012Do#012allow this access for now
 by executing:#012# ausearch -c 'httpd' --raw | audit2allow -M my-
httpd#012# semodule -X 300 -i my-httpd.pp#012
```

3.2. Run the suggested **sealert** command. Note the source context, the target objects, the policy, and the enforcing mode.

```
[root@serverb ~]# sealert -l 8824e73d-3ab0-4caf-8258-86e8792fee2d
SELinux is preventing /usr/sbin/httpd from getattr access on the file /
lab-content/lab.html.

*****  Plugin catchall (100. confidence) suggests
 **************************

If you believe that httpd should be allowed getattr access on the
 lab.html file by default.
Then you should report this as a bug.
You can generate a local policy module to allow this access.
Do
allow this access for now by executing:
# ausearch -c 'httpd' --raw | audit2allow -M my-httpd
# semodule -X 300 -i my-httpd.pp


Additional Information:
Source Context                system_u:system_r:httpd_t:s0
Target Context                unconfined_u:object_r:default_t:s0
Target Objects                /lab-content/lab.html [ file ]
Source                        httpd
Source Path                   /usr/sbin/httpd
Port                          <Unknown>
Host                          serverb.lab.example.com
Source RPM Packages
Target RPM Packages
Policy RPM                    selinux-policy-3.14.1-59.el8.noarch
Selinux Enabled               True
Policy Type                   targeted
Enforcing Mode                Enforcing
Host Name                     serverb.lab.example.com
```

```
Platform                      Linux serverb.lab.example.com
 4.18.0-67.el8.x86_64
                              #1 SMP Sat Feb 9 12:44:00 UTC 2019 x86_64
 x86_64
                              Alert Count                   2
                              First Seen                    2019-03-28
 15:19:46 CET
                              Last Seen                     2019-03-28
 15:19:46 CET
                              Local ID
 8824e73d-3ab0-4caf-8258-86e8792fee2d

                              Raw Audit Messages
                              type=AVC msg=audit(1553782786.213:864):
 avc:  denied  { getattr } for  pid=15606
 comm="httpd" path="/lab-content/lab.html" dev="vda1"
 ino=8763212 scontext=system_u:system_r:httpd_t:s0
 tcontext=unconfined_u:object_r:default_t:s0 tclass=file permissive=0


Hash: httpd,httpd_t,default_t,file,getattr
```

3.3. The **`Raw Audit Messages`** section of the **`sealert`** command contains information from the **`/var/log/audit/audit.log`**. Use the **`ausearch`** command to search the **`/var/log/audit/audit.log`** file. The **`-m`** option searches on the message type. The **`ts`** option searches based on time. This entry identifies the relevant process and file causing the alert. The process is the **`httpd`** Apache web server, the file is **`/lab-content/lab.html`**, and the context is **`system_r:httpd_t`**.

```
[root@serverb ~]# ausearch -m AVC -ts recent
time->Thu Mar 28 15:19:46 2019
type=PROCTITLE msg=audit(1553782786.213:864):
 proctitle=2F7573722F7362696E2F6874747064002D44464F524547524F554E44
type=SYSCALL msg=audit(1553782786.213:864): arch=c000003e
 syscall=6 success=no exit=-13 a0=7fb900004930 a1=7fb92dfca8e0
 a2=7fb92dfca8e0 a3=1 items=0 ppid=15491 pid=15606 auid=4294967295
 uid=48 gid=48 euid=48 suid=48 fsuid=48 egid=48 sgid=48 fsgid=48
 tty=(none) ses=4294967295 comm="httpd" exe="/usr/sbin/httpd"
 subj=system_u:system_r:httpd_t:s0 key=(null)
type=AVC msg=audit(1553782786.213:864): avc:  denied  { getattr }
 for  pid=15606 comm="httpd" path="/lab-content/lab.html"
 dev="vda1" ino=8763212 scontext=system_u:system_r:httpd_t:s0
 tcontext=unconfined_u:object_r:default_t:s0 tclass=file permissive=0
```

4. Display the SELinux context of the new HTTP document root and the original HTTP document root. Resolve the SELinux issue preventing Apache from serving web content.

   4.1. Use the **`ls -dZ`** to compare the document root of **`/lab-content`** and **`/var/www/html`**.

```
[root@serverb ~]# ls -dZ /lab-content /var/www/html
unconfined_u:object_r:default_t:s0 /lab-content/
 system_u:object_r:httpd_sys_content_t:s0 /var/www/html/
```

   4.2. Create a file context rule that sets the default type to **`httpd_sys_content_`** for **`/lab-content`** and all the files below it.

```
[root@serverb ~]# semanage fcontext -a -t httpd_sys_content_t '/lab-content(/.*)?'
```

   4.3. Use the **`restorecon`** command to set the SELinux context for the files in **`/lab-content`**.

```
[root@serverb ~]# restorecon -R /lab-content/
```

5. Verify that the SELinux issue has been resolved and Apache is able to serve web content.

   Use your web browser to refresh the `http://serverb/lab.html` link. Now you should see some web content.

```
This is the html file for the SELinux final lab on SERVERB.
```

6. Exit from `serverb`.

```
[root@serverb ~]# exit
logout
[student@serverb ~]$ exit
logout
Connection to serverb closed.
[student@workstation ~]$
```

Evaluation

On `workstation`, run the **`lab selinux-review grade`** script to confirm success on this exercise.

```
[student@workstation ~]$ lab selinux-review grade
```

Finish

On `workstation`, run the **`lab selinux-review finish`** script to complete the lab.

```
[student@workstation ~]$ lab selinux-review finish
```

This concludes the lab.

SUMMARY

In this chapter, you learned:

- The **`getenforce`** and **`setenforce`** commands are used to manage the SELinux mode of a system.
- The **`semanage`** command is used to manage SELinux policy rules. The **`restorecon`** command applies the context defined by the policy.
- Booleans are switches that change the behavior of the SELinux policy. They can be enabled or disabled and are used to tune the policy.
- The **`sealert`** displays useful information to help with SELinux troubleshooting.

CHAPTER 6

MANAGING BASIC STORAGE

GOAL

Create and manage storage devices, partitions, file systems, and swap spaces from the command line.

OBJECTIVES

- Create storage partitions, format them with file systems, and mount them for use.
- Create and manage swap spaces to supplement physical memory.

SECTIONS

- Adding Partitions, File Systems, and Persistent Mounts (and Guided Exercise)
- Managing Swap Space (and Guided Exercise)

LAB

Managing Basic Storage

ADDING PARTITIONS, FILE SYSTEMS, AND PERSISTENT MOUNTS

OBJECTIVES

After completing this section, you should be able to create storage partitions, format them with file systems, and mount them for use.

PARTITIONING A DISK

Disk partitioning allows system administrators to divide a hard drive into multiple logical storage units, referred to as partitions. By separating a disk into partitions, system administrators can use different partitions to perform different functions.

For example, disk partitioning is necessary or beneficial in these situations:

- Limit available space to applications or users.
- Separate operating system and program files from user files.
- Create a separate area for memory swapping.
- Limit disk space use to improve the performance of diagnostic tools and backup imaging.

MBR Partitioning Scheme

Since 1982, the *Master Boot Record (MBR)* partitioning scheme has dictated how disks are partitioned on systems running BIOS firmware. This scheme supports a maximum of four primary partitions. On Linux systems, with the use of extended and logical partitions, administrators can create a maximum of 15 partitions. Because partition size data is stored as 32-bit values, disks partitioned with the MBR scheme have a maximum disk and partition size of 2 TiB.

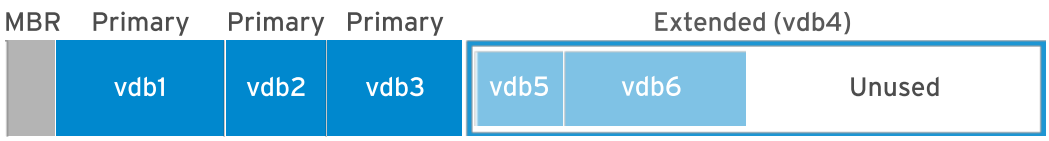

Figure 6.1: MBR Partitioning of the /dev/vdb storage device

Because physical disks are getting larger, and SAN-based volumes even larger than that, the 2 TiB disk and partition size limit of the MBR partitioning scheme is no longer a theoretical limit, but rather a real-world problem that system administrators encounter more and more frequently in production environments. As a result, the legacy MBR scheme is in the process of being superseded by the new *GUID Partition Table (GPT)* for disk partitioning.

GPT Partitioning Scheme

For systems running *Unified Extensible Firmware Interface (UEFI)* firmware, GPT is the standard for laying out partition tables on physical hard disks. GPT is part of the UEFI standard and addresses many of the limitations that the old MBR-based scheme imposes.

A GPT provides a maximum of 128 partitions. Unlike an MBR, which uses 32 bits for storing logical block addresses and size information, a GPT allocates 64 bits for logical block addresses. This

allows a GPT to accommodate partitions and disks of up to eight zebibytes (ZiB) or eight billion tebibytes.

In addition to addressing the limitations of the MBR partitioning scheme, a GPT also offers some additional features and benefits. A GPT uses a *globally unique identifier (GUID)* to identify each disk and partition. In contrast to an MBR, which has a single point of failure, a GPT offers redundancy of its partition table information. The primary GPT resides at the head of the disk, while a backup copy, the secondary GPT, is housed at the end of the disk. A GPT uses a checksum to detect errors and corruptions in the GPT header and partition table.

Figure 6.2: GPT Partitioning of the /dev/vdb storage device

MANAGING PARTITIONS WITH PARTED

Partition editors are programs which allow administrators to make changes to a disk's partitions, such as creating partitions, deleting partitions, and changing partition types. To perform these operations, administrators can use the Parted partition editor for both the MBR and the GPT partitioning scheme.

The **`parted`** command takes the device name of the whole disk as the first argument and one or more subcommands. The following example uses the **`print`** subcommand to display the partition table on the **`/dev/vda`** disk.

```
[root@host ~]# parted /dev/vda print
Model: Virtio Block Device (virtblk)
Disk /dev/vda: 53.7GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End     Size    Type     File system  Flags
 1      1049kB  10.7GB  10.7GB  primary  xfs          boot
 2      10.7GB  53.7GB  42.9GB  primary  xfs
```

If you do not provide a subcommand, **`parted`** opens an interactive session for issuing commands.

```
[root@host ~]# parted /dev/vda
GNU Parted 3.2
Using /dev/vda
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) print
Model: Virtio Block Device (virtblk)
Disk /dev/vda: 53.7GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End     Size    Type     File system  Flags
 1      1049kB  10.7GB  10.7GB  primary  xfs          boot
```

```
 2      10.7GB  53.7GB  42.9GB  primary  xfs

(parted) quit
```

By default, **parted** displays all the sizes in powers of 10 (KB, MB, GB). You can change that default with the **unit** subcommand which accepts the following parameters:

- **s** for sector
- **B** for byte
- **MiB**, **GiB**, or **TiB** (powers of 2)
- **MB**, **GB**, or **TB** (powers of 10)

```
[root@host ~]# parted /dev/vda unit s print
Model: Virtio Block Device (virtblk)
Disk /dev/vda: 104857600s
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start      End         Size       Type     File system  Flags
 1      2048s      20971486s   20969439s  primary  xfs          boot
 2      20971520s  104857535s  83886016s  primary  xfs
```

As shown in the example above, you can also specify multiple subcommands (here, **unit** and **print**) on the same line.

Writing the Partition Table on a New Disk

To partition a new drive, you first have to write a disk label to it. The disk label indicates which partitioning scheme to use.

> **NOTE**
> Keep in mind that **parted** makes the changes immediately. A mistake with **parted** could definitely lead to data loss.

As the `root` user, use the following command to write an MBR disk label to a disk.

```
[root@host ~]# parted /dev/vdb mklabel msdos
```

To write a GPT disk label, use the following command.

```
[root@host ~]# parted /dev/vdb mklabel gpt
```

> **WARNING**
> The **mklabel** subcommand wipes the existing partition table. Only use **mklabel** when the intent is to reuse the disk without regard to the existing data. If a new label changes the partition boundaries, all data in existing file systems will become inaccessible.

Creating MBR Partitions

Creating an MBR disk partition involves several steps:

1. Specify the disk device to create the partition on.

   As the `root` user, execute the **`parted`** command and specify the disk device name as an argument. This starts the **`parted`** command in interactive mode and displays a command prompt.

```
[root@host ~]# parted /dev/vdb
GNU Parted 3.2
Using /dev/vdb
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted)
```

2. Use the **`mkpart`** subcommand to create a new primary or extended partition.

```
(parted) mkpart
Partition type?  primary/extended? primary
```

> **NOTE**
> For situations where you need more than four partitions on an MBR-partitioned disk, create three primary partitions and one extended partition. This extended partition serves as a container within which you can create multiple logical partitions.

3. Indicate the file-system type that you want to create on the partition, such as **`xfs`** or **`ext4`**. This does not create the file system on the partition; it is only an indication of the partition type.

```
File system type?  [ext2]? xfs
```

   To get the list of the supported file-system types, use the following command:

```
[root@host ~]# parted /dev/vdb help mkpart
  mkpart PART-TYPE [FS-TYPE] START END     make a partition

    PART-TYPE is one of: primary, logical, extended
    FS-TYPE is one of: btrfs, nilfs2, ext4, ext3, ext2, fat32, fat16, hfsx,
    hfs+, hfs, jfs, swsusp, linux-swap(v1), linux-swap(v0), ntfs, reiserfs,
    hp-ufs, sun-ufs, xfs, apfs2, apfs1, asfs, amufs5, amufs4, amufs3,
    amufs2, amufs1, amufs0, amufs, affs7, affs6, affs5, affs4, affs3, affs2,
    affs1, affs0, linux-swap, linux-swap(new), linux-swap(old)
    START and END are disk locations, such as 4GB or 10%.  Negative values
    count from the end of the disk.  For example, -1s specifies exactly the
    last sector.

    'mkpart' makes a partition without creating a new file system on the
    partition.  FS-TYPE may be specified to set an appropriate partition
    ID.
```

4. Specify the sector on the disk that the new partition starts on.

```
Start? 2048s
```

Notice the **s** suffix to provide the value in sectors. You can also use the **MiB**, **GiB**, **TiB**, **MB**, **GB**, or **TB** suffixes. If you do not provide a suffix, **MB** is the default. **parted** may round the value you provide to satisfy disk constraints.

When **parted** starts, it retrieves the disk topology from the device. For example, the disk physical block size is typically a parameter that **parted** collects. With that information, **parted** ensures that the start position you provide correctly aligns the partition with the disk structure. Correct partition alignment is important for optimal performance. If the start position results in a misaligned partition, **parted** displays a warning. With most disks, a start sector that is a multiple of 2048 is a safe assumption.

5. Specify the disk sector where the new partition should end.

```
End? 1000MB
```

With **parted**, you cannot directly provide the size of your partition, but you can quickly compute it with the following formula:

```
Size = End - Start
```

As soon as you provide the end position, **parted** updates the partition table on the disk with the new partition details.

6. Exit **parted**.

```
(parted) quit
Information: You may need to update /etc/fstab.

[root@host ~]#
```

7. Run the **udevadm settle** command. This command waits for the system to detect the new partition and to create the associated device file under the **/dev** directory. It only returns when it is done.

```
[root@host ~]# udevadm settle
[root@host ~]#
```

As an alternative to the interactive mode, you can also create the partition as follows:

```
[root@host ~]# parted /dev/vdb mkpart primary xfs 2048s 1000MB
```

Creating GPT Partitions

The GTP scheme also uses the **parted** command to create new partitions:

1. Specify the disk device to create the partition on.

As the `root` user, execute the **`parted`** command with the disk device as the only argument to start **`parted`** in interactive mode with a command prompt.

```
[root@host ~]# parted /dev/vdb
GNU Parted 3.2
Using /dev/vdb
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted)
```

2. Use the **`mkpart`** subcommand to start creating the new partition.

   With the GPT scheme, each partition is given a name.

```
(parted) mkpart
Partition name?  []? usersdata
```

3. Indicate the file system type that you want to create on the partition, such as **`xfs`** or **`ext4`**. This does not create the file system on the partition; it is only an indication of the partition type.

```
File system type?  [ext2]? xfs
```

4. Specify the sector on the disk that the new partition starts on.

```
Start? 2048s
```

5. Specify the disk sector where the new partition should end.

```
End? 1000MB
```

   As soon as you provide the end position, **`parted`** updates the partition table on the disk with the new partition details.

6. Exit **`parted`**.

```
(parted) quit
Information: You may need to update /etc/fstab.

[root@host ~]#
```

7. Run the **`udevadm settle`** command. This command waits for the system to detect the new partition and to create the associated device file under the **`/dev`** directory. It only returns when it is done.

```
[root@host ~]# udevadm settle
[root@host ~]#
```

As an alternative to the interactive mode, you can also create the partition as follows:

```
[root@host ~]# parted /dev/vdb mkpart usersdata xfs 2048s 1000MB
```

Deleting Partitions

The following steps apply for both the MBR and GPT partitioning schemes.

1. Specify the disk that contains the partition to be removed.

   As the `root` user, execute the **`parted`** command with the disk device as the only argument to start **`parted`** in interactive mode with a command prompt.

```
[root@host ~]# parted /dev/vdb
GNU Parted 3.2
Using /dev/vdb
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted)
```

2. Identify the partition number of the partition to delete.

```
(parted) print
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 5369MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size   File system  Name       Flags
 1      1049kB  1000MB  999MB  xfs          usersdata
```

3. Delete the partition.

```
(parted) rm 1
```

   The **`rm`** subcommand immediately deletes the partition from the partition table on the disk.

4. Exit **`parted`**.

```
(parted) quit
Information: You may need to update /etc/fstab.

[root@host ~]#
```

CREATING FILE SYSTEMS

After the creation of a block device, the next step is to add a file system to it. Red Hat Enterprise Linux supports many different file system types, but two common ones are XFS and ext4. Anaconda, the installer for Red Hat Enterprise Linux, uses XFS by default.

As `root`, use the **`mkfs.xfs`** command to apply an XFS file system to a block device. For ext4, use **`mkfs.ext4`**.

```
[root@host ~]# mkfs.xfs /dev/vdb1
meta-data=/dev/vdb1              isize=512    agcount=4, agsize=60992 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=1        finobt=1, sparse=1, rmapbt=0
```

```
         =                       reflink=1
data     =                       bsize=4096   blocks=243968, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0, ftype=1
log      =internal log           bsize=4096   blocks=1566, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
```

MOUNTING FILE SYSTEMS

After you have added the file system, the last step is to mount the file system to a directory in the directory structure. When you mount a file system onto the directory hierarchy, user-space utilities can access or write files on the device.

Manually Mounting File Systems

Administrators use the **mount** command to manually attach the device onto a directory location, or mount point. The **mount** command expects the device, the mount point, and optionally file system options as arguments. The file-system options customize the behavior of the file system.

```
[root@host ~]# mount /dev/vdb1 /mnt
```

You also use the **mount** command to view currently mounted file systems, the mount points, and the options.

```
[root@host ~]# mount | grep vdb1
/dev/vdb1 on /mnt type xfs (rw,relatime,seclabel,attr2,inode64,noquota)
```

Persistently Mounting File Systems

Manually mounting a file system is a good way to verify that a formatted device is accessible and working as expected. However, when the server reboots, the system does not automatically mount the file system onto the directory tree again; the data is intact on the file system, but users cannot access it.

To make sure that the system automatically mounts the file system at system boot, add an entry to the **/etc/fstab** file. This configuration file lists the file systems to mount at system boot.

/etc/fstab is a white-space-delimited file with six fields per line.

```
[root@host ~]# cat /etc/fstab

#
# /etc/fstab
# Created by anaconda on Wed Feb 13 16:39:59 2019
#
# Accessible filesystems, by reference, are maintained under '/dev/disk/'.
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info.
#
# After editing this file, run 'systemctl daemon-reload' to update systemd
# units generated from this file.
```

```
#
UUID=a8063676-44dd-409a-b584-68be2c9f5570   /        xfs   defaults   0 0
UUID=7a20315d-ed8b-4e75-a5b6-24ff9e1f9838   /dbdata  xfs   defaults   0 0
```

When you add or remove an entry in the **/etc/fstab** file, run the **systemctl daemon-reload** command, or reboot the server, for `systemd` to register the new configuration.

```
[root@host ~]# systemctl daemon-reload
```

The *first field* specifies the device. This example uses the UUID to specify the device. File systems create and store the UUID in their super block at creation time. Alternatively, you could use the device file, such as **/dev/vdb1**.

NOTE

Using the UUID is preferable because block device identifiers can change in certain scenarios, such as a cloud provider changing the underlying storage layer of a virtual machine, or the disks being detected in a different order with each system boot. The block device file name may change, but the UUID remains constant in the file system's super block.

Use the **lsblk --fs** command to scan the block devices connected to a machine and retrieve the file system UUIDs.

```
[root@host ~]# lsblk --fs
NAME   FSTYPE LABEL UUID                                 MOUNTPOINT
sr0
vda
└─vda1 xfs          a8063676-44dd-409a-b584-68be2c9f5570 /
vdb
└─vdb1 xfs          7a20315d-ed8b-4e75-a5b6-24ff9e1f9838 /dbdata
```

The *second field* is the directory mount point, from which the block device will be accessible in the directory structure. The mount point must exist; if not, create it with the **mkdir** command.

The *third field* contains the file-system type, such as **xfs** or **ext4**.

The *fourth field* is the comma-separated list of options to apply to the device. **defaults** is a set of commonly used options. The **mount**(8) man page documents the other available options.

The *fifth field* is used by the **dump** command to back up the device. Other backup applications do not usually use this field.

The *last field*, the **fsck** order field, determines if the **fsck** command should be run at system boot to verify that the file systems are clean. The value in this field indicates the order in which **fsck** should run. For XFS file systems, set this field to **0** because XFS does not use **fsck** to check its file-system status. For ext4 file systems, set it to **1** for the root file system and **2** for the other ext4 file systems. This way, **fsck** processes the root file system first and then checks file systems on separate disks concurrently, and file systems on the same disk in sequence.

NOTE

Having an incorrect entry in **`/etc/fstab`** may render the machine non-bootable. Administrators should verify that the entry is valid by unmounting the new file system and using **`mount /mountpoint`**, which reads **`/etc/fstab`**, to remount the file system. If the **`mount`** command returns an error, correct it before rebooting the machine.

As an alternative, you can use the **`findmnt --verify`** command to control the **`/etc/fstab`** file.

REFERENCES

`info parted` (*GNU Parted User Manual*)

`parted`(8), **`mkfs`**(8), **`mount`**(8), **`lsblk`**(8), and **`fstab`**(5) man pages

For more information, refer to the *Configuring and managing file systems* guide at https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/configuring_and_managing_file_systems/

GUIDED EXERCISE

ADDING PARTITIONS, FILE SYSTEMS, AND PERSISTENT MOUNTS

In this exercise, you will create a partition on a new storage device, format it with an XFS file system, configure it to be mounted at boot, and mount it for use.

OUTCOMES

You should be able to use **`parted`**, **`mkfs.xfs`**, and other commands to create a partition on a new disk, format it, and persistently mount it.

BEFORE YOU BEGIN

Log in as the `student` user on `workstation` using `student` as the password.

On `workstation`, run the **`lab storage-partitions start`** command. This command runs a start script that determines if the `servera` machine is reachable on the network. It also prepares the second disk on `servera` for the exercise.

```
[student@workstation ~]$ lab storage-partitions start
```

1. Use the **`ssh`** command to log in to `servera` as the `student` user. The systems are configured to use SSH keys for authentication, therefore a password is not required.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

2. Use the **`sudo -i`** command to switch to the `root` user. If prompted, use `student` as the password.

```
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

3. Use **`parted`** to create a new disk label of type **`msdos`** on the **`/dev/vdb`** disk to prepare that new disk for the MBR partitioning scheme.

```
[root@servera ~]# parted /dev/vdb mklabel msdos
Information: You may need to update /etc/fstab.
```

4. Add a new primary partition that is 1 GB in size. For proper alignment, start the partition at the sector 2048. Set the partition file system type to XFS.

4.1. Use **parted** interactive mode to help you create the partition.

```
[root@servera ~]# parted /dev/vdb
GNU Parted 3.2
Using /dev/vdb
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) mkpart
Partition type?  primary/extended? primary
File system type?  [ext2]? xfs
Start? 2048s
End? 1001MB
(parted) quit
Information: You may need to update /etc/fstab.
```

Because the partition starts at the sector 2048, the previous command sets the end position to 1001MB to get a partition size of 1000MB (1 GB).

As an alternative, you can perform the same operation with the following noninteractive command: **parted /dev/vdb mkpart primary xfs 2048s 1001MB**

4.2. Verify your work by listing the partitions on **/dev/vdb**.

```
[root@servera ~]# parted /dev/vdb print
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 5369MB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End     Size    Type     File system  Flags
 1      1049kB  1001MB  1000MB  primary
```

4.3. Run the **udevadm settle** command. This command waits for the system to register the new partition and returns when it is done.

```
[root@servera ~]# udevadm settle
[root@servera ~]#
```

5. Format the new partition with the XFS file system.

```
[root@servera ~]# mkfs.xfs /dev/vdb1
meta-data=/dev/vdb1              isize=512    agcount=4, agsize=61056 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=1        finobt=1, sparse=1, rmapbt=0
         =                       reflink=1
data     =                       bsize=4096   blocks=244224, imaxpct=25
         =                       sunit=0      swidth=0 blks
```

```
naming   =version 2              bsize=4096   ascii-ci=0, ftype=1
log      =internal log           bsize=4096   blocks=1566, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
```

6. Configure the new file system to mount at **/archive** persistently.

6.1. Use **mkdir** to create the **/archive** directory mount point.

```
[root@servera ~]# mkdir /archive
[root@servera ~]#
```

6.2. Use the **lsblk** command with the **--fs** option to discover the UUID of the **/dev/vdb1** device.

```
[root@servera ~]# lsblk --fs /dev/vdb
NAME   FSTYPE LABEL UUID                                 MOUNTPOINT
vdb
└─vdb1 xfs          e3db1abe-6d96-4faa-a213-b96a6f85dcc1
```

The UUID in the previous output is probably different on your system.

6.3. Add an entry to **/etc/fstab**. In the following command, replace the UUID with the one you discovered from the previous step.

```
[root@servera ~]# vim /etc/fstab
...output omitted...
UUID=e3db1abe-6d96-4faa-a213-b96a6f85dcc1   /archive   xfs   defaults
  0   0
```

6.4. Update `systemd` for the system to register the new **/etc/fstab** configuration.

```
[root@servera ~]# systemctl daemon-reload
[root@servera ~]#
```

6.5. Execute the **mount /archive** command to mount the new file system using the new entry added to **/etc/fstab**.

```
[root@servera ~]# mount /archive
[root@servera ~]#
```

6.6. Verify that the new file system is mounted at **/archive**.

```
[root@servera ~]# mount | grep /archive
/dev/vdb1 on /archive type xfs
 (rw,relatime,seclabel,attr2,inode64,noquota)
```

7. Reboot `servera`. After the server has rebooted, log in and verify that **`/dev/vdb1`** is mounted at **`/archive`**. When done, log off from `servera`.

   7.1. Reboot `servera`.

```
[root@servera ~]# systemctl reboot
Connection to servera closed by remote host.
Connection to servera closed.
[student@workstation ~]$
```

   7.2. Wait a few minutes for `servera` to reboot and log in as the `student` user.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

   7.3. Verify that **`/dev/vdb1`** is mounted at **`/archive`**.

```
[student@servera ~]$ mount | grep /archive
/dev/vdb1 on /archive type xfs
 (rw,relatime,seclabel,attr2,inode64,noquota)
```

   7.4. Log off from `servera`.

```
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```

Finish

On `workstation`, run the **`lab storage-partitions finish`** script to complete this exercise.

```
[student@workstation ~]$ lab storage-partitions finish
```

This concludes the guided exercise.

MANAGING SWAP SPACE

OBJECTIVES

After completing this section, you should be able to create and manage swap spaces to supplement physical memory.

INTRODUCING SWAP SPACE CONCEPTS

A *swap space* is an area of a disk under the control of the Linux kernel memory management subsystem. The kernel uses swap space to supplement the system RAM by holding inactive pages of memory. The combined system RAM plus swap space is called *virtual memory*.

When the memory usage on a system exceeds a defined limit, the kernel searches through RAM looking for idle memory pages assigned to processes. The kernel writes the idle pages to the swap area and reassigns the RAM pages to other processes. If a program requires access to a page on disk, the kernel locates another idle page of memory, writes it to disk, then recalls the needed page from the swap area.

Because swap areas reside on disk, swap is slow when compared with RAM. While it is used to augment system RAM, you should not consider swap space as a sustainable solution for insufficient RAM for your workload.

Sizing the Swap Space

Administrators should size the swap space based on the memory workload on the system. Application vendors sometimes provide recommendations on that subject. The following table provides some guidance based on the total amount of physical memory.

RAM and Swap Space Recommendations

RAM	SWAP SPACE	SWAP SPACE IF ALLOWING FOR HIBERNATION
2 GiB or less	Twice the RAM	Three times the RAM
Between 2 GiB and 8 GiB	Same as RAM	Twice the RAM
Between 8 GiB and 64 GiB	At least 4 GiB	1.5 times the RAM
More than 64 GiB	At least 4 GiB	Hibernation is not recommended

The laptop and desktop hibernation function uses the swap space to save the RAM contents before powering off the system. When you turn the system back on, the kernel restores the RAM contents from the swap space and does not need a complete boot. For those systems, the swap space needs to be greater than the amount of RAM.

The Knowledgebase article in the Reference section at the end of this section gives more guidance on sizing the swap space.

CREATING A SWAP SPACE

To create a swap space, you need to perform the following:

- Create a partition with a file system type of **linux-swap**.
- Place a swap signature on the device.

Creating a Swap Partition

Use **parted** to create a partition of the desired size and set its file system type to **linux-swap**. In the past, tools looked at the partition file system type to determine if the device should be activated; however, that is no longer the case. Even though utilities no longer use the partition file system type, setting that type allows administrators to quickly determine the partition's purpose.

The following example creates a 256 MB partition.

```
[root@host ~]# parted /dev/vdb
GNU Parted 3.2
Using /dev/vdb
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) print
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 5369MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size    File system  Name  Flags
 1      1049kB  1001MB  1000MB               data

(parted) mkpart
Partition name?  []? swap1
File system type?  [ext2]? linux-swap
Start? 1001MB
End? 1257MB
(parted) print
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 5369MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size    File system     Name   Flags
 1      1049kB  1001MB  1000MB                  data
 2      1001MB  1257MB  256MB   linux-swap(v1)  swap1

(parted) quit
Information: You may need to update /etc/fstab.
```

After creating the partition, run the **udevadm settle** command. This command waits for the system to detect the new partition and to create the associated device file in **/dev**. It only returns when it is done.

```
[root@host ~]# udevadm settle
[root@host ~]#
```

Formatting the Device

The **mkswap** command applies a swap signature to the device. Unlike other formatting utilities, **mkswap** writes a single block of data at the beginning of the device, leaving the rest of the device unformatted so the kernel can use it for storing memory pages.

```
[root@host ~]# mkswap /dev/vdb2
Setting up swapspace version 1, size = 244 MiB (255848448 bytes)
no label, UUID=39e2667a-9458-42fe-9665-c5c854605881
```

ACTIVATING A SWAP SPACE

You can use the **swapon** command to activate a formatted swap space.

Use **swapon** with the device as a parameter, or use **swapon -a** to activate all the swap spaces listed in the **/etc/fstab** file. Use the **swapon --show** and **free** commands to inspect the available swap spaces.

```
[root@host ~]# free
              total        used        free      shared  buff/cache   available
Mem:        1873036      134688     1536436       16748      201912     1576044
Swap:             0           0           0
[root@host ~]# swapon /dev/vdb2
[root@host ~]# free
              total        used        free      shared  buff/cache   available
Mem:        1873036      135044     1536040       16748      201952     1575680
Swap:        249852           0      249852
```

You can deactivate a swap space using the **swapoff** command. If the swap space has pages written to it, **swapoff** tries to move those pages to other active swap spaces or back into memory. If it cannot write data to other places, the **swapoff** command fails with an error, and the swap space stays active.

Activating Swap Space Persistently

To activate a swap space at every boot, place an entry in the **/etc/fstab** file. The example below shows a typical line in **/etc/fstab** based on the swap space created above.

```
UUID=39e2667a-9458-42fe-9665-c5c854605881   swap   swap   defaults   0 0
```

The example uses the UUID as the *first field*. When you format the device, the **mkswap** command displays that UUID. If you lost the output of **mkswap**, use the **lsblk --fs** command. As an alternative, you can also use the device name in the first field.

The *second field* is typically reserved for the mount point. However, for swap devices, which are not accessible through the directory structure, this field takes the placeholder value **swap**.

The *third field* is the file system type. The file system type for swap space is **swap**.

The *fourth field* is for options. The example uses the **defaults** option. The **defaults** option includes the mount option **auto**, which means activate the swap space automatically at system boot.

The final two fields are the **dump** flag and **fsck** order. Swap spaces require neither backing up nor file-system checking and so these fields should be set to zero.

When you add or remove an entry in the **/etc/fstab** file, run the **systemctl daemon-reload** command, or reboot the server, for systemd to register the new configuration.

```
[root@host ~]# systemctl daemon-reload
```

Setting the Swap Space Priority

By default, the system uses swap spaces in series, meaning that the kernel uses the first activated swap space until it is full, then it starts using the second swap space. However, you can define a priority for each swap space to force that order.

To set the priority, use the **pri** option in **/etc/fstab**. The kernel uses the swap space with the highest priority first. The default priority is -2.

The following example shows three swap spaces defined in **/etc/fstab**. The kernel uses the last entry first, with **pri=10**. When that space is full, it uses the second entry, with **pri=4**. Finally, it uses the first entry, which has a default priority of -2.

```
UUID=af30cbb0-3866-466a-825a-58889a49ef33   swap   swap   defaults  0 0
UUID=39e2667a-9458-42fe-9665-c5c854605881   swap   swap   pri=4     0 0
UUID=fbd7fa60-b781-44a8-961b-37ac3ef572bf   swap   swap   pri=10    0 0
```

Use **swapon --show** to display the swap space priorities.

When swap spaces have the same priority, the kernel writes to them in a round-robin fashion.

REFERENCES

mkswap(8), **swapon**(8), **swapoff**(8), **mount**(8), and **parted**(8) man pages

Knowledgebase: What is the recommended swap size for Red Hat platforms?
https://access.redhat.com/solutions/15244

GUIDED EXERCISE

MANAGING SWAP SPACE

In this exercise, you will create and format a partition for use as swap space, format it as swap, and activate it persistently.

OUTCOMES

You should be able to create a partition and a swap space on a disk using the GPT partitioning scheme.

BEFORE YOU BEGIN

Log in as the `student` user on `workstation` using `student` as the password.

On `workstation`, run the **`lab storage-swap start`** command. This command runs a start script that determines if the `servera` machine is reachable on the network. It also prepares the second disk on `servera` for the exercise.

```
[student@workstation ~]$ lab storage-swap start
```

1. Use the **`ssh`** command to log in to `servera` as the `student` user. The systems are configured to use SSH keys for authentication, therefore a password is not required.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

2. Use the **`sudo -i`** command to switch to the `root` user. If prompted, use `student` as the password.

```
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

3. Use the **`parted`** command to inspect the **`/dev/vdb`** disk.

```
[root@servera ~]# parted /dev/vdb print
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 5369MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size    File system  Name  Flags
 1      1049kB  1001MB  1000MB               data
```

Notice that the disk already has a partition table and uses the GPT partitioning scheme. Also, a 1 GB partition already exists.

4. Add a new partition that is 500 MB in size for use as swap space. Set the partition type to **linux-swap**.

   4.1. Use **parted** to create the partition. Because the disk uses the GPT partitioning scheme, you need to give a name to the partition. Call it **myswap**.

```
[root@servera ~]# parted /dev/vdb mkpart myswap linux-swap 1001MB
 1501MB
Information: You may need to update /etc/fstab.
```

   Notice in the previous command that the start position, 1001 MB, is the end of the existing first partition. This way **parted** makes sure that the new partition immediately follows the previous one, without any gap.

   Because the partition starts at the 1001 MB position, the command sets the end position to 1501 MB to get a partition size of 500 MB.

   4.2. Verify your work by listing the partitions on **/dev/vdb**.

```
[root@servera ~]# parted /dev/vdb print
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 5369MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size    File system  Name   Flags
 1      1049kB  1001MB  1000MB               data
```

   The size of the new partition is not exactly 500 MB. This is because **parted** has to align the partition with the disk layout.

   4.3. Run the **udevadm settle** command. This command waits for the system to register the new partition and returns when it is done.

```
[root@servera ~]# udevadm settle
[root@servera ~]#
```

5. Initialize the newly created partition as swap space.

```
[root@servera ~]# mkswap /dev/vdb2
Setting up swapspace version 1, size = 476 MiB (499118080 bytes)
no label, UUID=cb7f71ca-ee82-430e-ad4b-7dda12632328
```

6. Enable the newly created swap space.

6.1. Use the **swapon --show** command to show that creating and initializing swap space does not yet enable it for use.

```
[root@servera ~]# swapon --show
[root@servera ~]#
```

6.2. Enable the newly created swap space.

```
[root@servera ~]# swapon /dev/vdb2
[root@servera ~]#
```

6.3. Verify that the newly created swap space is now available.

```
[root@servera ~]# swapon --show
NAME      TYPE      SIZE USED PRIO
/dev/vdb2 partition 476M   0B   -2
```

6.4. Disable the swap space.

```
[root@servera ~]# swapoff /dev/vdb2
[root@servera ~]#
```

6.5. Confirm that the swap space is disabled.

```
[root@servera ~]# swapon --show
[root@servera ~]#
```

7. Configure the new swap space to be enabled at system boot.

7.1. Use the **lsblk** command with the **--fs** option to discover the UUID of the **/dev/vdb2** device.

```
[root@servera ~]# lsblk --fs /dev/vdb2
NAME FSTYPE LABEL UUID                                 MOUNTPOINT
vdb2 swap         cb7f71ca-ee82-430e-ad4b-7dda12632328
```

The UUID in the previous output is probably different on your system.

7.2. Add an entry to **/etc/fstab**. In the following command, replace the UUID with the one you discovered from the previous step.

```
[root@servera ~]# vim /etc/fstab
...output omitted...
UUID=cb7f71ca-ee82-430e-ad4b-7dda12632328   swap   swap   defaults   0
   0
```

7.3. Update `systemd` for the system to register the new **/etc/fstab** configuration.

```
[root@servera ~]# systemctl daemon-reload
[root@servera ~]#
```

7.4. Enable the swap space using the entry just added to **/etc/fstab**.

```
[root@servera ~]# swapon -a
[root@servera ~]#
```

7.5. Verify that the new swap space is enabled.

```
[root@servera ~]# swapon --show
NAME      TYPE      SIZE USED PRIO
/dev/vdb2 partition 476M   0B   -2
```

8. Reboot `servera`. After the server has rebooted, log in and verify that the swap space is enabled. When done, log off from `servera`.

   8.1. Reboot `servera`.

```
[root@servera ~]# systemctl reboot
Connection to servera closed by remote host.
Connection to servera closed.
[student@workstation ~]$
```

   8.2. Wait a few minutes for `servera` to reboot and log in as the `student` user.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

   8.3. Verify that the swap space is enabled.

```
[root@servera ~]# swapon --show
NAME      TYPE      SIZE USED PRIO
/dev/vdb2 partition 476M   0B   -2
```

   8.4. Log off from `servera`.

```
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```

Finish

On `workstation`, run the **`lab storage-swap finish`** script to complete this exercise.

```
[student@workstation ~]$ lab storage-swap finish
```

This concludes the guided exercise.

LAB

MANAGING BASIC STORAGE

PERFORMANCE CHECKLIST

In this lab, you will create several partitions on a new disk, formatting some with file systems and mounting them, and activating others as swap spaces.

OUTCOMES

You should be able to:

- Display and create partitions using the **`parted`** command.
- Create new file systems on partitions and persistently mount them.
- Create swap spaces and activate them at boot.

BEFORE YOU BEGIN

Log in to `workstation` as `student` using `student` as the password.

On `workstation`, run the **`lab storage-review start`** command. This command runs a start script that determines if the `serverb` machine is reachable on the network. It also prepares the second disk on `serverb` for the exercise.

```
[student@workstation ~]$ lab storage-review start
```

1. New disks are available on `serverb`. On the first new disk, create a 2 GB GPT partition named **`backup`**. Because it may be difficult to set the exact size, a size between 1.8 GB and 2.2 GB is acceptable. Set the correct file-system type on that partition to host an XFS file system.

   The password for the `student` user account on `serverb` is `student`. This user has full `root` access through **`sudo`**.
2. Format the 2 GB partition with an XFS file system and persistently mount it at **`/backup`**.
3. On the same new disk, create two 512 MB GPT partitions named **`swap1`** and **`swap2`**. A size between 460 MB and 564 MB is acceptable. Set the correct file-system type on those partitions to host swap spaces.
4. Initialize the two 512 MiB partitions as swap spaces and configure them to activate at boot. Set the swap space on the **`swap2`** partition to be preferred over the other.
5. To verify your work, reboot `serverb`. Confirm that the system automatically mounts the first partition at **`/backup`**. Also, confirm that the system activates the two swap spaces.

   When done, log off from `serverb`.

Evaluation

On `workstation`, run the **`lab storage-review grade`** script to confirm success on this exercise.

```
[student@workstation ~]$ lab storage-review grade
```

Finish

On `workstation`, run the **`lab storage-review finish`** script to complete the lab.

```
[student@workstation ~]$ lab storage-review finish
```

This concludes the lab.

SOLUTION

MANAGING BASIC STORAGE

PERFORMANCE CHECKLIST

In this lab, you will create several partitions on a new disk, formatting some with file systems and mounting them, and activating others as swap spaces.

OUTCOMES

You should be able to:

- Display and create partitions using the **`parted`** command.
- Create new file systems on partitions and persistently mount them.
- Create swap spaces and activate them at boot.

BEFORE YOU BEGIN

Log in to `workstation` as `student` using `student` as the password.

On `workstation`, run the **`lab storage-review start`** command. This command runs a start script that determines if the `serverb` machine is reachable on the network. It also prepares the second disk on `serverb` for the exercise.

```
[student@workstation ~]$ lab storage-review start
```

1. New disks are available on `serverb`. On the first new disk, create a 2 GB GPT partition named **`backup`**. Because it may be difficult to set the exact size, a size between 1.8 GB and 2.2 GB is acceptable. Set the correct file-system type on that partition to host an XFS file system.

   The password for the `student` user account on `serverb` is `student`. This user has full `root` access through **`sudo`**.

   1.1. Use the **`ssh`** command to log in to `serverb` as the `student` user. The systems are configured to use SSH keys for authentication, therefore a password is not required.

   ```
   [student@workstation ~]$ ssh student@serverb
   ...output omitted...
   [student@serverb ~]$
   ```

   1.2. Because creating partitions and file systems requires `root` access, use the **`sudo -i`** command to switch to the `root` user. If prompted, use `student` as the password.

   ```
   [student@serverb ~]$ sudo -i
   [sudo] password for student: student
   [root@serverb ~]#
   ```

1.3. Use the **lsblk** command to identify the new disks. Those disks should not have any partitions yet.

```
[root@serverb ~]# lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sr0     11:0    1 1024M  0 rom
vda    252:0    0   10G  0 disk
└vda1 252:1    0   10G  0 part /
vdb    252:16   0    5G  0 disk
vdc    252:32   0    5G  0 disk
vdd    252:48   0    5G  0 disk
```

Notice that the first new disk, **vdb**, does not have any partitions.

1.4. Confirm that the disk has no label.

```
[root@serverb ~]# parted /dev/vdb print
Error: /dev/vdb: unrecognised disk label
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 5369MB
Sector size (logical/physical): 512B/512B
Partition Table: unknown
Disk Flags:
```

1.5. Use **parted** and the **mklabel** subcommand to define the GPT partitioning scheme.

```
[root@serverb ~]# parted /dev/vdb mklabel gpt
Information: You may need to update /etc/fstab.
```

1.6. Create the 2 GB partition. Name it **backup** and set its type to **xfs**. Start the partition at sector 2048.

```
[root@serverb ~]# parted /dev/vdb mkpart backup xfs 2048s 2GB
Information: You may need to update /etc/fstab.
```

1.7. Confirm the correct creation of the new partition.

```
[root@serverb ~]# parted /dev/vdb print
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 5369MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size    File system  Name    Flags
 1      1049kB  2000MB  1999MB               backup
```

1.8. Run the **udevadm settle** command. This command waits for the system to detect the new partition and to create the **/dev/vdb1** device file. It only returns when it is done.

```
[root@serverb ~]# udevadm settle
[root@serverb ~]#
```

2. Format the 2 GB partition with an XFS file system and persistently mount it at **/backup**.

2.1. Use the **mkfs.xfs** command to format the **/dev/vbd1** partition.

```
[root@serverb ~]# mkfs.xfs /dev/vdb1
meta-data=/dev/vdb1              isize=512    agcount=4, agsize=121984
 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=1        finobt=1, sparse=1,
 rmapbt=0
         =                       reflink=1
data     =                       bsize=4096   blocks=487936, imaxpct=25
         =                       sunit=0      swidth=0 blks
```

```
naming   =version 2              bsize=4096   ascii-ci=0, ftype=1
log      =internal log           bsize=4096   blocks=2560, version=2
         =                       sectsz=512   sunit=0 blks, lazy-
count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
```

2.2. Create the **/backup** mount point.

```
[root@serverb ~]# mkdir /backup
[root@serverb ~]#
```

2.3. Before adding the new file system to **/etc/fstab**, retrieve its UUID.

```
[root@serverb ~]# lsblk --fs /dev/vdb1
NAME FSTYPE LABEL UUID                                 MOUNTPOINT
vdb1 xfs          a3665c6b-4bfb-49b6-a528-74e268b058dd
```

The UUID on your system is probably different.

2.4. Edit **/etc/fstab** and define the new file system.

```
[root@serverb ~]# vim /etc/fstab
...output omitted...
UUID=a3665c6b-4bfb-49b6-a528-74e268b058dd   /backup   xfs   defaults
 0 0
```

2.5. Force `systemd` to reread the **/etc/fstab** file.

```
[root@serverb ~]# systemctl daemon-reload
[root@serverb ~]#
```

2.6. Manually mount **/backup** to verify your work. Confirm that the mount is successful.

```
[root@serverb ~]# mount /backup
[root@serverb ~]# mount | grep /backup
/dev/vdb1 on /backup type xfs
 (rw,relatime,seclabel,attr2,inode64,noquota)
```

3. On the same new disk, create two 512 MB GPT partitions named **swap1** and **swap2**. A size between 460 MB and 564 MB is acceptable. Set the correct file-system type on those partitions to host swap spaces.

3.1. Retrieve the end position of the first partition by displaying the current partition table on **/dev/vdb**. In the next step, you use that value as the start of the **swap1** partition.

```
[root@serverb ~]# parted /dev/vdb print
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 5369MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
```

```
Disk Flags:

Number  Start   End     Size    File system  Name    Flags
 1      1049kB  2000MB  1999MB  xfs          backup
```

3.2. Create the first 512 MB partition named **swap1**. Set its type to **linux-swap**. Use the end position of the first partition as the starting point. The end position is 2000 MB + 512 MB = 2512 MB

```
[root@serverb ~]# parted /dev/vdb mkpart swap1 linux-swap 2000MB 2512M
Information: You may need to update /etc/fstab.
```

3.3. Create the second 512 MB partition named **swap2**. Set its type to **linux-swap**. Use the end position of the previous partition as the starting point: **2512M**. The end position is 2512 MB + 512 MB = 3024 MB

```
[root@serverb ~]# parted /dev/vdb mkpart swap2 linux-swap 2512M 3024M
Information: You may need to update /etc/fstab.
```

3.4. Display the partition table to verify your work.

```
[root@serverb ~]# parted /dev/vdb print
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 5369MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size    File system  Name    Flags
 1      1049kB  2000MB  1999MB  xfs          backup
 2      2000MB  2512MB  513MB                swap1   swap
```

3.5. Run the **udevadm settle** command. This command waits for the system to register the new partitions and to create the device files.

```
[root@serverb ~]# udevadm settle
[root@serverb ~]#
```

4. Initialize the two 512 MiB partitions as swap spaces and configure them to activate at boot. Set the swap space on the **swap2** partition to be preferred over the other.

4.1. Use the **mkswap** command to initialize the swap partitions.

```
[root@serverb ~]# mkswap /dev/vdb2
Setting up swapspace version 1, size = 489 MiB (512749568 bytes)
no label, UUID=87976166-4697-47b7-86d1-73a02f0fc803
[root@serverb ~]# mkswap /dev/vdb3
Setting up swapspace version 1, size = 488 MiB (511700992 bytes)
no label, UUID=4d9b847b-98e0-4d4e-9ef7-dfaaf736b942
```

Take note of the UUIDs of the two swap spaces. You use that information in the next step. If you cannot see the **mkswap** output anymore, use the **lsblk --fs** command to retrieve the UUIDs.

4.2. Edit **/etc/fstab** and define the new swap spaces. To set the swap space on the **swap2** partition to be preferred over **swap1**, give it a higher priority with the **pri** option.

```
[root@serverb ~]# vim /etc/fstab
...output omitted...
UUID=a3665c6b-4bfb-49b6-a528-74e268b058dd   /backup xfs   defaults  0 0
UUID=87976166-4697-47b7-86d1-73a02f0fc803   swap    swap  pri=10    0 0
UUID=4d9b847b-98e0-4d4e-9ef7-dfaaf736b942   swap    swap  pri=20    0 0
```

4.3. Force `systemd` to reread the **/etc/fstab** file.

```
[root@serverb ~]# systemctl daemon-reload
[root@serverb ~]#
```

4.4. Use the **swapon -a** command to activate the new swap spaces. Use the **swapon --show** command to confirm the correct activation of the swap spaces.

```
[root@serverb ~]# swapon -a
[root@serverb ~]# swapon --show
NAME      TYPE      SIZE USED PRIO
/dev/vdb2 partition 489M   0B   10
/dev/vdb3 partition 488M   0B   20
```

5. To verify your work, reboot `serverb`. Confirm that the system automatically mounts the first partition at **`/backup`**. Also, confirm that the system activates the two swap spaces.

When done, log off from `serverb`.

5.1. Reboot `serverb`.

```
[root@serverb ~]# systemctl reboot
[root@serverb ~]#
Connection to serverb closed by remote host.
Connection to serverb closed.
[student@workstation ~]$
```

5.2. Wait a few minutes for `serverb` to reboot and then log in as the `student` user.

```
[student@workstation ~]$ ssh student@serverb
...output omitted...
[student@serverb ~]$
```

5.3. Verify that the system automatically mounts **`/dev/vdb1`** at **`/backup`**.

```
[student@serverb ~]$ mount | grep /backup
/dev/vdb1 on /backup type xfs
 (rw,relatime,seclabel,attr2,inode64,noquota)
```

5.4. Use the **`swapon --show`** command to confirm that the system activates both swap spaces.

```
[student@serverb ~]$ swapon --show
NAME      TYPE      SIZE USED PRIO
/dev/vdb2 partition 489M   0B   10
/dev/vdb3 partition 488M   0B   20
```

5.5. Log off from `serverb`.

```
[student@serverb ~]$ exit
logout
Connection to serverb closed.
[student@workstation ~]$
```

Evaluation

On `workstation`, run the **`lab storage-review grade`** script to confirm success on this exercise.

```
[student@workstation ~]$ lab storage-review grade
```

Finish

On `workstation`, run the **`lab storage-review finish`** script to complete the lab.

```
[student@workstation ~]$ lab storage-review finish
```

This concludes the lab.

SUMMARY

In this chapter, you learned:

- You use the **`parted`** command to add, modify, and remove partitions on disks with the MBR or the GPT partitioning scheme.
- You use the **`mkfs.xfs`** command to create XFS file systems on disk partitions.
- You need to add file-system mount commands to **`/etc/fstab`** to make those mounts persistent.
- You use the **`mkswap`** command to initialize swap spaces.

CHAPTER 7

MANAGING LOGICAL VOLUMES

GOAL

Create and manage logical volumes containing file systems and swap spaces from the command line.

OBJECTIVES

- Create and manage logical volumes from storage devices, and format them with file systems or prepare them with swap spaces.
- Add and remove storage assigned to volume groups, and nondestructively extend the size of a logical volume formatted with a file system.

SECTIONS

- Creating Logical Volumes (and Guided Exercise)
- Extending Logical Volumes (and Guided Exercise)

LAB

Managing Logical Volumes

CREATING LOGICAL VOLUMES

OBJECTIVES

After completing this section, you should be able to:

- Describe logical volume management components and concepts.
- Implement LVM storage.
- Display LVM component information.

LOGICAL VOLUME MANAGEMENT (LVM) CONCEPTS

Logical volumes and logical volume management make it easier to manage disk space. If a file system that hosts a logical volume needs more space, it can be allocated to its logical volume from the free space in its volume group and the file system can be resized. If a disk starts to fail, a replacement disk can be registered as a physical volume with the volume group and the logical volume's extents can be migrated to the new disk.

LVM Definitions

Physical devices
: Physical devices are the storage devices used to save data stored in a logical volume. These are block devices and could be disk partitions, whole disks, RAID arrays, or SAN disks. A device must be initialized as an LVM physical volume in order to be used with LVM. The entire device will be used as a physical volume.

Physical volumes (PVs)
: You must initialize a device as a physical volume before using it in an LVM system. LVM tools segment physical volumes into *physical extents (PEs)*, which are small chunks of data that act as the smallest storage block on a physical volume.

Volume groups (VGs)
: Volume groups are storage pools made up of one or more physical volumes. This is the functional equivalent of a whole disk in basic storage. A PV can only be allocated to a single VG. A VG can consist of unused space and any number of logical volumes.

Logical volumes (LVs)
: Logical volumes are created from free physical extents in a volume group and provide the "storage" device used by applications, users, and the operating system. LVs are a collection of *logical extents (LEs)*, which map to physical extents, the smallest storage chunk of a PV. By default, each LE maps to one PE. Setting specific LV options changes this mapping; for example, mirroring causes each LE to map to two PEs.

IMPLEMENTING LVM STORAGE

Creating LVM storage requires several steps. The first step is to determine which physical devices to use. After a set of suitable devices have been assembled, they are initialized as physical volumes so that they are recognized as belonging to LVM. The physical volumes are then combined into a volume group. This creates a pool of disk space out of which logical volumes can be allocated.

Logical volumes created from the available space in a volume group can be formatted with a file system, activated as swap space, and mounted or activated persistently.

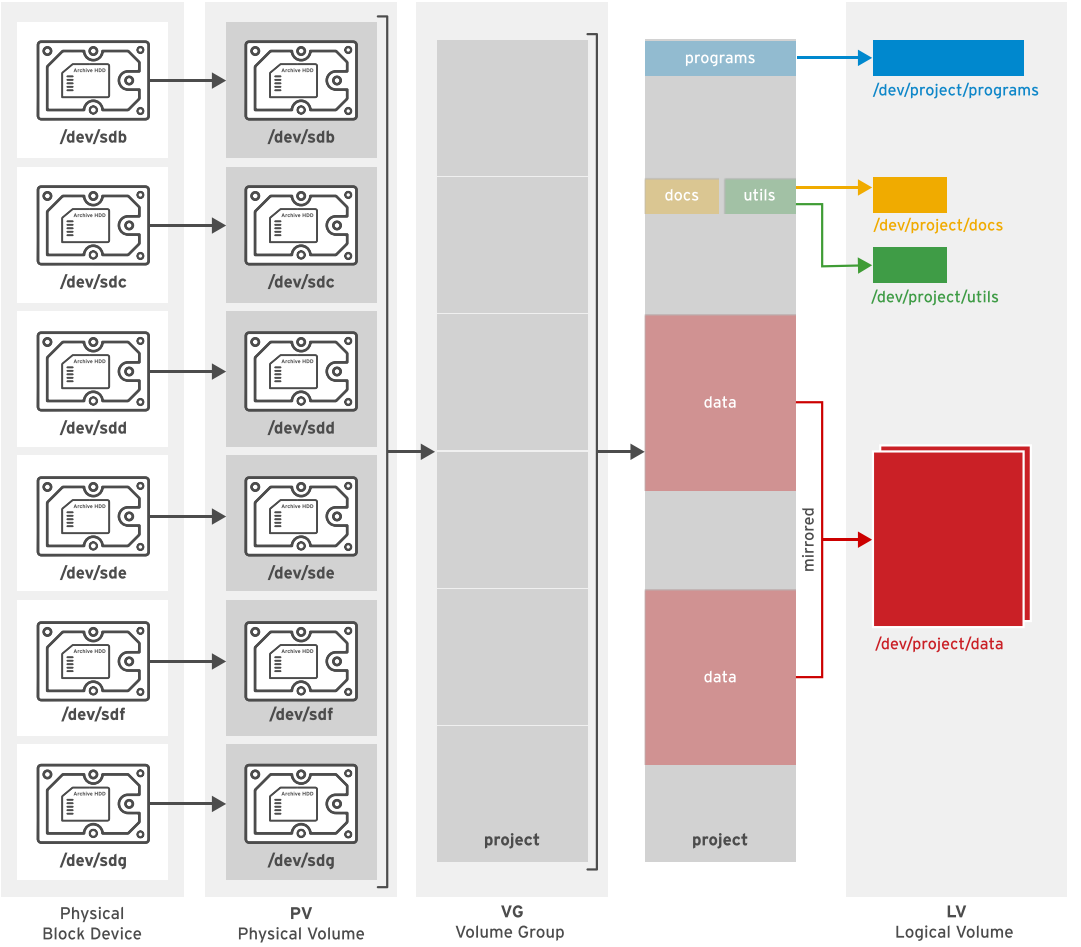

Figure 7.1: Logical volume management components

LVM provides a comprehensive set of command-line tools for implementing and managing LVM storage. These command-line tools can be used in scripts, making them suitable for automation.

> **IMPORTANT**
> The following examples use device **vdb** and its partitions to illustrate LVM commands. In practice, these examples would need to use the correct devices for the disk and disk partitions that are being used by the system. Use the **`lsblk`**, **`blkid`**, or **`cat /proc/partitions`** commands to identify the devices on your system.

Creating a Logical Volume

To create a logical volume, perform the following steps:

Prepare the physical device.

Use **`parted`**, **`gdisk`**, or **`fdisk`** to create a new partition for use with LVM. Always set the partition type to **`Linux LVM`** on LVM partitions; use **`0x8e`** for MBR partitions. If necessary, use **`partprobe`** to register the new partition with the kernel.

Alternatively, use a whole disk, a RAID array, or a SAN disk.

A physical device only needs to be prepared if there are none prepared already and a new physical volume is required to create or extend a volume group.

```
[root@host ~]# parted -s /dev/vdb mkpart primary 1MiB 769MiB
[root@host ~]# parted -s /dev/vdb mkpart primary 770MiB 1026MiB
[root@host ~]# parted -s /dev/vdb set 1 lvm on
[root@host ~]# parted -s /dev/vdb set 2 lvm on
```

Create a physical volume.

Use **pvcreate** to label the partition (or other physical device) as a physical volume. The **pvcreate** command divides the physical volume into physical extents (PEs) of a fixed size, for example, 4 MiB blocks. You can label multiple devices at the same time by using space-delimited device names as arguments to **pvcreate**.

```
[root@host ~]# pvcreate /dev/vdb2 /dev/vdb1
```

This labels the devices **/dev/vdb2** and **/dev/vdb1** as PVs, ready for allocation into a volume group.

A PV only needs to be created if there are no PVs free to create or extend a VG.

Create a volume group.

Use **vgcreate** to collect one or more physical volumes into a volume group. A volume group is the functional equivalent of a hard disk; you will create logical volumes from the pool of free physical extents in the volume group.

The **vgcreate** command-line consists of a volume group name followed by one or more physical volumes to allocate to this volume group.

```
[root@host ~]# vgcreate vg01 /dev/vdb2 /dev/vdb1
```

This creates a VG called `vg01` that is the combined size, in PE units, of the two PVs **/dev/vdb2** and **/dev/vdb1**.

A VG only needs to be created if none already exist. Additional VGs may be created for administrative reasons to manage the use of PVs and LVs. Otherwise, existing VGs can be extended to accommodate new LVs when needed.

Create a logical volume.

Use **lvcreate** to create a new logical volume from the available physical extents in a volume group. At a minimum, the **lvcreate** command includes the **-n** option to set the LV name, either the **-L** option to set the LV size in bytes or the **-l** option to set the LV size in extents, and the name of the volume group hosting this logical volume.

```
[root@host ~]# lvcreate -n lv01 -L 700M vg01
```

This creates an LV called `lv01`, 700 MiB in size, in the VG `vg01`. This command will fail if the volume group does not have a sufficient number of free physical extents for the requested size.

Note also that the size will be rounded to a factor of the physical extent size if the size cannot match exactly.

You can specify the size using the **-L** option, which expects sizes in bytes, mebibytes (binary megabytes, 1048576 bytes), gibibytes (binary gigabytes), or similar. Alternatively, you can use the **-l** option, which expects sizes specified as a number of physical extents.

The following list provides some examples of creating LVs:

- **lvcreate -L 128M**: Size the logical volume to exactly 128 MiB.
- **lvcreate -l 128** : Size the logical volume to exactly 128 extents. The total number of bytes depends on the size of the physical extent block on the underlying physical volume.

> **IMPORTANT**
> Different tools display the logical volume name using either the traditional name, **/dev/*vgname/lvname***, or the kernel device mapper name, **/dev/mapper/*vgname-lvname***.

Add the file system.

Use **mkfs** to create an **XFS** file system on the new logical volume. Alternatively, create a file system based on your preferred file system, for example, **ext4**.

```
[root@host ~]# mkfs -t xfs /dev/vg01/lv01
```

To make the file system available across reboots, perform the following steps:

- Use **mkdir** to create a mount point.

```
[root@host ~]# mkdir /mnt/data
```

- Add an entry to the **/etc/fstab** file:

```
/dev/vg01/lv01  /mnt/data xfs  defaults 1 2
```

> **NOTE**
> Mounting a logical volume by name is equivalent to mounting by UUID because LVM finds its physical volumes based on a UUID even if you initially add them to the volume group by name.

- Run **mount /mnt/data** to mount the file system that you just added in **/etc/fstab**.

```
[root@host ~]# mount /mnt/data
```

Removing a Logical Volume

To remove *all* logical volume components, perform the following steps:

Prepare the file system.

Move all data that must be kept to another file system. Use the **umount** command to unmount the file system and then remove any **/etc/fstab** entries associated with this file system.

```
[root@host ~]# umount /mnt/data
```

> **WARNING**
> Removing a logical volume destroys any data stored on the logical volume. Back up or move your data *before* you remove the logical volume.

Remove the logical volume.

Use **lvremove *DEVICE_NAME*** to remove a logical volume that is no longer needed.

```
[root@host ~]# lvremove /dev/vg01/lv01
```

Unmount the LV file system before running this command. The command prompts for confirmation before removing the LV.

The LV's physical extents are freed and made available for assignment to existing or new LVs in the volume group.

Remove the volume group.

Use **vgremove *VG_NAME*** to remove a volume group that is no longer needed.

```
[root@host ~]# vgremove vg01
```

The VG's physical volumes are freed and made available for assignment to existing or new VGs on the system.

Remove the physical volumes.

Use **pvremove** to remove physical volumes that are no longer needed. Use a space-delimited list of PV devices to remove more than one at a time. This command deletes the PV metadata from the partition (or disk). The partition is now free for reallocation or reformatting.

```
[root@host ~]# pvremove /dev/vdb2 /dev/vdb1
```

REVIEWING LVM STATUS INFORMATION

Physical Volumes

Use **pvdisplay** to display information about physical volumes. To list information about all physical volumes, use the command without arguments. To list information about a specific physical volume, pass that device name to the command.

```
[root@host ~]# pvdisplay /dev/vdb1
  --- Physical volume ---
  PV Name               /dev/vdb1                    (1)
  VG Name               vg01                         (2)
```

```
  PV Size               768.00 MiB / not usable 4.00 MiB  (3)
  Allocatable           yes

  PE Size               4.00 MiB                          (4)
  Total PE              191

  Free PE               16                                (5)
  Allocated PE          175
  PV UUID               JWzDpn-LG3e-n2oi-9Etd-VT2H-PMem-1ZXwP1
```

1. **`PV Name`** maps to the device name.
2. **`VG Name`** shows the volume group where the PV is allocated.
3. **`PV Size`** shows the physical size of the PV, including any unusable space.
4. **`PE Size`** is the physical extent size, which is the smallest size a logical volume can be allocated.

   It is also the multiplying factor when calculating the size of any value reported in PE units, such as *Free PE*; for example: 26 PEs x 4 MiB (the *PE Size*) equals 104 MiB of free space. A logical volume size is rounded to a factor of PE units.

   LVM sets the PE size automatically, although it is possible to specify it.
5. **`Free PE`** shows how many PE units are available for allocation to new logical volumes.

Volume Groups

Use **`vgdisplay`** to display information about volume groups. To list information about all volume groups, use the command without arguments. To list information about a specific volume group, pass that VG name to the command.

```
[root@host ~]# vgdisplay vg01
  --- Volume group ---

  VG Name               vg01                (1)
  System ID
  Format                lvm2
  Metadata Areas        2
  Metadata Sequence No  2
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                1
  Open LV               1
  Max PV                0
  Cur PV                2
  Act PV                2

  VG Size               1016.00 MiB         (2)
  PE Size               4.00 MiB

  Total PE              254                 (3)
  Alloc PE / Size       175 / 700.00 MiB

  Free  PE / Size       79 / 316.00 MiB     (4)
  VG UUID               3snNw3-CF71-CcYG-Llk1-p6EY-rHEv-xfUSez
```

1. **`VG Name`** is the name of the volume group.
2. **`VG Size`** is the total size of the storage pool available for logical volume allocation.

3. **`Total PE`** is the total size expressed in PE units.
4. **`Free PE / Size`** shows how much space is free in the VG for allocating to new LVs or to extend existing LVs.

Logical Volumes

Use **`lvdisplay`** to display information about logical volumes. If you provide no argument to the command, it displays information about all LVs; if you provide an LV device name as an argument, the command displays information about that specific device.

```
[root@host ~]# lvdisplay /dev/vg01/lv01
  --- Logical volume ---
  LV Path                /dev/vg01/lv01          1
  LV Name                lv01
  VG Name                vg01                    2
  LV UUID                5IyRea-W8Zw-xLHk-3h2a-IuVN-YaeZ-i3IRrN
  LV Write Access        read/write
  LV Creation host, time host.lab.example.com, 2019-03-28 17:17:47 -0400
  LV Status              available
  # open                 1
  LV Size                700 MiB                 3
  Current LE             175                     4
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - current set to       256
  Block device           252:0
```

1. **`LV Path`** shows the device name of the logical volume.

   Some tools may report the device name as **`/dev/mapper/`*`vgname-lvname`***; both represent the same LV.
2. **`VG Name`** shows the volume group that the LV is allocated from.
3. **`LV Size`** shows the total size of the LV. Use file-system tools to determine the free space and used space for storage of data.
4. **`Current LE`** shows the number of logical extents used by this LV. An LE usually maps to a physical extent in the VG, and therefore the physical volume.

REFERENCES

`lvm`(8), **`pvcreate`**(8), **`vgcreate`**(8), **`lvcreate`**(8), **`pvremove`**(8), **`vgremove`**(8), **`lvremove`**(8), **`pvdisplay`**(8), **`vgdisplay`**(8), **`lvdisplay`**(8), **`fdisk`**(8), **`gdisk`**(8), **`parted`**(8), **`partprobe`**(8), and **`mkfs`**(8) man pages

GUIDED EXERCISE

CREATING LOGICAL VOLUMES

In this lab, you will create a physical volume, volume group, logical volume, and an XFS file system. You will also persistently mount the logical volume file system.

OUTCOMES

You should be able to:

- Create physical volumes, volume groups, and logical volumes with LVM tools.
- Create new file systems on logical volumes and persistently mount them.

BEFORE YOU BEGIN

Log in as the `student` user on `workstation` using `student` as the password.

On `workstation`, run the **`lab lvm-creating start`** command. This command runs a start script that determines if the `servera` machine is reachable on the network. It also verifies that storage is available and that the appropriate software packages are installed.

```
[student@workstation ~]$ lab lvm-creating start
```

1. Use the **`ssh`** command to log in to `servera` as the `student` user. The systems are configured to use SSH keys for authentication, therefore a password is not required.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

2. Use the **`sudo -i`** command to switch to the `root` user. The password for the `student` user is `student`.

```
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

3. Create the physical resources.

   3.1. Use **`parted`** to create two 256 MiB partitions and set them to type Linux LVM.

```
[root@servera ~]# parted -s /dev/vdb mklabel gpt
[root@servera ~]# parted -s /dev/vdb mkpart primary 1MiB 257MiB
[root@servera ~]# parted -s /dev/vdb set 1 lvm on
[root@servera ~]# parted -s /dev/vdb mkpart primary 258MiB 514MiB
[root@servera ~]# parted -s /dev/vdb set 2 lvm on
```

3.2. Use **udevadm settle** for the system to register the new partitions.

```
[root@servera ~]# udevadm settle
```

4. Use **pvcreate** to add the two new partitions as PVs.

```
[root@servera ~]# pvcreate /dev/vdb1 /dev/vdb2
  Physical volume "/dev/vdb1" successfully created.
  Physical volume "/dev/vdb2" successfully created.
```

5. Use **vgcreate** to create a new VG named `servera_01_vg` built from the two PVs.

```
[root@servera ~]# vgcreate servera_01_vg /dev/vdb1 /dev/vdb2
  Volume group "servera_01_vg" successfully created
```

6. Use **lvcreate** to create a 400 MiB LV named `servera_01_lv` from the `servera_01_vg` VG.

```
[root@servera ~]# lvcreate -n servera_01_lv -L 400M servera_01_vg
  Logical volume "servera_01_lv" created.
```

This creates a device named **/dev/servera_01_vg/servera_01_lv** but without a file system on it.

7. Add a persistent file system.

7.1. Add an XFS file system on the `servera_01_lv` LV with the **`mkfs`** command.

```
[root@servera ~]# mkfs -t xfs /dev/servera_01_vg/servera_01_lv
...output omitted...
```

7.2. Create a mount point at **`/data`**.

```
[root@servera ~]# mkdir /data
```

7.3. Add the following line to the end of **`/etc/fstab`** on `servera`:

```
/dev/servera_01_vg/servera_01_lv    /data    xfs  defaults  1 2
```

7.4. Use **`systemctl daemon-reload`** to update `systemd` with the new **`/etc/fstab`** configuration.

```
[root@servera ~]# systemctl daemon-reload
```

7.5. Verify the **`/etc/fstab`** entry and mount the new `servera_01_lv` LV device with the **`mount`** command.

```
[root@servera ~]# mount /data
```

8. Test and review your work.

8.1. As a final test, copy some files to **`/data`** and verify how many were copied.

```
[root@servera ~]# cp -a /etc/*.conf /data
[root@servera ~]# ls /data | wc -l
34
```

You will verify that you still have the same number of files in the next guided exercise.

8.2. **`parted /dev/vdb print`** lists the partitions that exist on **`/dev/vdb`**.

```
[root@servera ~]# parted /dev/vdb print
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 5369MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:
```

```
Number  Start   End     Size    File system  Name     Flags
 1      1049kB  269MB   268MB               primary  lvm
 2      271MB   539MB   268MB               primary  lvm
```

Notice the **Number** column, which contains the values **1** and **2**. These correspond to **/dev/vdb1** and **/dev/vdb2**, respectively. Also notice the **Flags** column, which indicates the partition type.

8.3. **pvdisplay** displays information about each of the physical volumes. Optionally, include the device name to limit details to a specific PV.

```
[root@servera ~]# pvdisplay /dev/vdb2
  --- Physical volume ---
  PV Name               /dev/vdb2
  VG Name               servera_01_vg
  PV Size               256.00 MiB / not usable 4.00 MiB
  Allocatable           yes
  PE Size               4.00 MiB
```

```
  Total PE              63
  Free PE               26
  Allocated PE          37
  PV UUID               2z0Cf3-99YI-w9ny-alEW-wWhL-S8RJ-M2rfZk
```

This shows that the PV is allocated to VG `servera_01_vg`, is 256 MiB in size (although 4 MiB is not usable), and the physical extent size (**`PE Size`**) is 4 MiB (the smallest allocatable LV size).

There are 63 PEs, of which 26 are free for allocation to LVs in the future and 37 are currently allocated to LVs. These translate to MiB values as follows:

- Total 252 MiB (63 PEs x 4 MiB); remember, 4 MiB is unusable.
- Free 104 MiB (26 PEs x 4 MiB)
- Allocated 148 MiB (37 PEs x 4 MiB)

8.4. **`vgdisplay vgname`** shows information about the volume group named `vgname`.

```
[root@servera ~]# vgdisplay servera_01_vg
```

Verify the following values:

- **`VG Size`** is **`504.00MiB`**.
- **`Total PE`** is **`126`**.
- **`Alloc PE / Size`** is **`100 / 400.00MiB`**.
- **`Free PE / Size`** is **`26 / 104.00MiB`**.

8.5. **`lvdisplay /dev/vgname/lvname`** displays information about the logical volume named `lvname`.

```
[root@servera ~]# lvdisplay /dev/servera_01_vg/servera_01_lv
```

Review the **`LV Path`**, **`LV Name`**, **`VG Name`**, **`LV Status`**, **`LV Size`**, and **`Current LE`** (logical extents, which map to physical extents).

8.6. The **`mount`** command shows all mounted devices and any mount options. It should include **`/dev/servera_01_vg/servera_01_lv`**.

> **NOTE**
> Many tools report the device mapper name instead, **`/dev/mapper/servera_01_vg-servera_01_lv`**; it is the same logical volume.

```
[root@servera ~]# mount
```

You should see (probably on the last line) **`/dev/mapper/servera_01_vg-servera_01_lv`** mounted on **`/data`** and the associated mount information.

8.7. **df -h** displays human-readable disk free space. Optionally, include the mount point to limit details to that file system.

```
[root@servera ~]# df -h /data
Filesystem                                  Size  Used Avail Use%
 Mounted on
/dev/mapper/servera_01_vg-servera_01_lv  395M   24M  372M   6% /data
```

Allowing for file-system metadata, these values are expected.

9. Log off from `servera`.

```
[root@servera ~]# exit
logout
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```

Finish

On `workstation`, run the **lab lvm-creating finish** script to finish this exercise. This script removes the storage configured on `servera` during the exercise.

```
[student@workstation ~]$ lab lvm-creating finish
```

This concludes the guided exercise.

EXTENDING LOGICAL VOLUMES

OBJECTIVES

After completing this section, you should be able to:

- Extend a volume group (VG) using **`pvcreate`** and **`vgextend`**, and use **`vgdisplay`** to verify the results.
- Reduce a VG using **`pvmove`** and **`vgreduce`**.
- Extend a logical volume (LV) using **`lvextend`**.
- Resize **XFS** file systems with **`xfs_growfs`**.
- Resize **`ext4`** file systems with **`resize2fs`**.

EXTENDING AND REDUCING A VOLUME GROUP

You can add more disk space to a volume group by adding additional physical volumes. This is called *extending the volume group*. Then, you can assign the new physical extents from the additional physical volumes to logical volumes.

You can remove unused physical volumes from a volume group. This is called *reducing the volume group*. First, use the **`pvmove`** command to move data from extents on one physical volume to extents on other physical volumes in the volume group. In this way, a new disk can be added to an existing volume group, data can be moved from an older or slower disk to a new disk, and the old disk removed from the volume group. You can perform these actions while the logical volumes in the volume group are in use.

IMPORTANT
The following examples use the device **vdb** and its partitions to illustrate LVM commands. In practice, use the appropriate devices for the disk and disk partitions on your own system.

Extending a Volume Group

To extend a volume group, perform the following steps:

Prepare the physical device and create the physical volume.

As with creating a new volume group, you must create and prepare a new partition for use as a physical volume if there are none prepared already.

```
[root@host ~]# parted -s /dev/vdb mkpart primary 1027MiB 1539MiB
[root@host ~]# parted -s /dev/vdb set 3 lvm on
[root@host ~]# pvcreate /dev/vdb3
```

A PV only needs to be created if there are no PVs free to extend the VG.

Extend the volume group.

Use **vgextend** to add the new physical volume to the volume group. Use the VG name and PV device name as arguments to **vgextend**.

```
[root@host ~]# vgextend vg01 /dev/vdb3
```

This extends the vg01 VG by the size of the **/dev/vdb3** PV.

Verify that the new space is available.

Use **vgdisplay** to confirm the additional physical extents are available. Inspect the **Free PE / Size** in the output. It should not be zero.

```
[root@host ~]# vgdisplay vg01
  --- Volume group ---
  VG Name               vg01
...output omitted...
  Free  PE / Size       178 / 712.00 MiB
...output omitted...
```

Reducing a Volume Group

To reduce a volume group, perform the following steps:

Move the physical extents.

Use **pvmove *PV_DEVICE_NAME*** to relocate any physical extents from the physical volume you want to remove to other physical volumes in the volume group. The other physical volumes must have a sufficient number of free extents to accommodate this move. This is only possible if there are enough free extents in the VG and if all of those come from other PVs.

```
[root@host ~]# pvmove /dev/vdb3
```

This command moves the PEs from **/dev/vdb3** to other PVs with free PEs in the same VG.

> **WARNING**
> Before using **pvmove**, back up data stored on all logical volumes in the volume group. An unexpected power loss during the operation may leave the volume group in an inconsistent state. This could cause loss of data on logical volumes in the volume group.

Reduce the volume group.

Use **vgreduce *VG_NAME PV_DEVICE_NAME*** to remove a physical volume from a volume group.

```
[root@host ~]# vgreduce vg01 /dev/vdb3
```

This removes the **/dev/vdb3** PV from the vg01 VG and it can now be added to another VG. Alternatively, **pvremove** can be used to permanently stop using the device as a PV.

EXTENDING A LOGICAL VOLUME AND XFS FILE SYSTEM

One benefit of logical volumes is the ability to increase their size without experiencing downtime. Free physical extents in a volume group can be added to a logical volume to extend its capacity, which can then be used to extend the file system it contains.

Extending a Logical Volume

To extend a logical volume, perform the following steps:

Verify that the volume group has space available.

Use **`vgdisplay`** to verify that there are sufficient physical extents available.

```
[root@host ~]# vgdisplay vg01
  --- Volume group ---
  VG Name               vg01
...output omitted...
  Free  PE / Size       178 / 712.00 MiB
...output omitted...
```

Inspect the **`Free PE / Size`** in the output. Confirm that the volume group has sufficient free space for the LV extension. If insufficient space is available, then extend the volume group appropriately. See the section called "Extending and Reducing a Volume Group".

Extend the logical volume.

Use **`lvextend`***`LV_DEVICE_NAME`* to extend the logical volume to a new size.

```
[root@host ~]# lvextend -L +300M /dev/vg01/lv01
```

This increases the size of the logical volume `lv01` by 300 MiB. Notice the plus sign (+) in front of the size, which means add this value to the existing size; otherwise, the value defines the final size of the LV.

As with **`lvcreate`**, different methods exist to specify the size: the **`-l`** option expects the number of physical extents as the argument. The **`-L`** option expects sizes in bytes, mebibytes, gibibytes, and similar.

The following list provides some examples of extending LVs.

Extending LVs Examples

COMMAND	RESULTS
`lvextend -l 128`	Resize the logical volume to *exactly* 128 extents in size.
`lvextend -l +128`	Add 128 extents to the current size of the logical volume.
`lvextend -L 128M`	Resize the logical volume to *exactly* 128 MiB.
`lvextend -L +128M`	Add 128 MiB to the current size of the logical volume.

COMMAND	RESULTS
lvextend -l +50%FREE	Add 50 percent of the current free space in the VG to the LV.

Extend the file system.

Use **xfs_growfs *mountpoint*** to expand the file system to occupy the extended LV. The target file system must be mounted when you use the **xfs_growfs** command. You can continue to use the file system while it is being resized.

```
[root@host ~]# xfs_growfs /mnt/data
```

NOTE
A common mistake is to run **lvextend** but to forget to run **xfs_growfs**. An alternative to running the two steps consecutively is to include the **-r** option with the **lvextend** command. This resizes the file system after the LV is extended, using **fsadm**(8). It works with a number of different file systems.

- Verify the new size of the mounted file system:

  df -h /*mountpoint*.

EXTENDING A LOGICAL VOLUME AND EXT4 FILE SYSTEM

The steps for extending an **ext4**-based logical volume are essentially the same as for an **XFS**-based LV, except for the step that resizes the file system. Review the section called "Extending a Logical Volume and XFS File System".

Verify that the volume group has space available.

Use **vgdisplay *VGNAME*** to verify that the volume group has a sufficient number of physical extents available.

Extend the logical volume.

Use **lvextend -l +*extents* /dev/*vgname/lvname*** to extend the logical volume */dev/vgname/lvname* by the *extents* value.

Extend the file system.

Use **resize2fs /dev/*vgname/lvname*** to expand the file system to occupy the new extended LV. The file system can be mounted and in use while the extension command is running. You can include the **-p** option to monitor the progress of the resize operation.

```
[root@host ~]# resize2fs /dev/vg01/lv01
```

NOTE
The primary difference between **xfs_growfs** and **resize2fs** is the argument passed to identify the file system. **xfs_growfs** takes the mount point and **resize2fs** takes the logical volume name.

EXTEND A LOGICAL VOLUME AND SWAP SPACE

Logical volumes formatted as swap space can be extended as well, however the process is different than the one for extending a file system, such as **`ext4`** or **`XFS`**. Logical volumes formatted with a file system can be extended dynamically with no downtime. Logical volumes formatted with swap space must be taken offline in order to extend them.

Verify the volume group has space available.

Use **`vgdisplay`** ***`vgname`*** to verify that a sufficient number of free physical extents are available.

Deactivate the swap space.

Use **`swapoff -v /dev/`*`vgname/lvname`*** to deactivate the swap space on the logical volume.

> **WARNING**
> Your system must have enough free memory or swap space to accept anything that needs to page in when the swap space on the logical volume is deactivated.

Extend the logical volume.

`lvextend -l +`*`extents`*` /dev/`*`vgname/lvname`* extends the logical volume */dev/vgname/lvname* by the *extents* value.

Format the logical volume as swap space.

`mkswap /dev/`*`vgname/lvname`* formats the entire logical volume as swap space.

Activate the swap space.

Use **`swapon -va /dev/`*`vgname/lvname`*** to activate the swap space on the logical volume.

> **REFERENCES**
> **`lvm`**(8), **`pvcreate`**(8), **`pvmove`**(8), **`vgdisplay`**(8), **`vgextend`**(8), **`vgreduce`**(8), **`vgdisplay`**(8), **`vgextend`**(8), **`vgreduce`**(8), **`lvextend`**(8), **`fdisk`**(8), **`gdisk`**(8), **`parted`**(8), **`partprobe`**(8), **`xfs_growfs`**(8), and **`resize2fs`**(8) **`swapoff`**(8) **`swapon`**(8) **`mkswap`**(8)man pages

GUIDED EXERCISE

EXTENDING LOGICAL VOLUMES

In this lab, you will extend the logical volume added in the previous practice exercise.

OUTCOMES

You should be able to:

- Extend the volume group to include an additional physical volume.
- Resize the logical volume while the file system is still mounted and in use.

BEFORE YOU BEGIN

Log in as the `student` user on `workstation` using `student` as the password.

On `workstation`, run the **`lab lvm-extending start`** command. This command runs a start script that determines if the host `servera` is reachable on the network and ensures that the storage from the previous guided exercise is available.

```
[student@workstation ~]$ lab lvm-extending start
```

1. Use the **`ssh`** command to log in to `servera` as the `student` user.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

2. Use the **`sudo -i`** command to switch to `root` at the shell prompt.

```
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

3. Use **`vgdisplay`** to determine if the VG has sufficient free space to extend the LV to a total size of 700 MiB.

```
[root@servera ~]# vgdisplay servera_01_vg
  --- Volume group ---
  VG Name               servera_01_vg
  System ID
  Format                lvm2
...output omitted...
  VG Size               504.00 MiB
  PE Size               4.00 MiB
  Total PE              126
```

```
  Alloc PE / Size       100 / 400.00 MiB
  Free  PE / Size       26 / 104.00 MiB
  VG UUID               OBBAtU-2nBS-4SW1-khmF-yJzi-z7bD-DpCrAV
```

Only 104 MiB is available (26 PEs x 4 MiB extents) and you need at least 300 MiB to have 700 MiB in total. You need to extend the VG.

For later comparison, use **df** to record current disk free space:

```
[root@servera ~]# df -h /data
Filesystem                              Size  Used Avail Use% Mounted on
/dev/mapper/servera_01_vg-servera_01_lv  395M   24M  372M   6% /data
```

4. Create the physical resource.

   4.1. Use **parted** to create an additional partition of 512 MiB and set it to type Linux LVM.

```
[root@servera ~]# parted -s /dev/vdb mkpart primary 515MiB 1027MiB
[root@servera ~]# parted -s /dev/vdb set 3 lvm on
```

   4.2. Use **udevadm settle** for the system to register the new partition.

```
[root@servera ~]# udevadm settle
```

5. Use **pvcreate** to add the new partition as a PV.

```
[root@servera ~]# pvcreate /dev/vdb3
  Physical volume "/dev/vdb3" successfully created.
```

6. Extend the volume group.

   6.1. Use **vgextend** to extend the VG named `servera_01_vg`, using the new `/dev/vdb3` PV.

```
[root@servera ~]# vgextend servera_01_vg /dev/vdb3
  Volume group "servera_01_vg" successfully extended
```

   6.2. Use **vgdisplay** to inspect the `servera_01_vg` VG free space again. There should be plenty of free space now.

```
[root@servera ~]# vgdisplay servera_01_vg
  --- Volume group ---
  VG Name               servera_01_vg
  System ID
  Format                lvm2
...output omitted...
  VG Size               1012.00 MiB
  PE Size               4.00 MiB
```

```
  Total PE              253
  Alloc PE / Size       100 / 400.00 MiB
  Free  PE / Size       153 / 612.00 MiB
  VG UUID               OBBAtU-2nBS-4SW1-khmF-yJzi-z7bD-DpCrAV
```

612 MiB of free space is now available (153 PEs x 4 MiB extents).

7. Use **lvextend** to extend the existing LV to 700 MiB.

```
[root@servera ~]# lvextend -L 700M /dev/servera_01_vg/servera_01_lv
  Size of logical volume servera_01_vg/servera_01_lv changed from 400.00 MiB
 (100 extents) to 700.00 MiB (175 extents).
    Logical volume servera_01_vg/servera_01_lv successfully resized.
```

NOTE

The example specifies the exact size to make the final LV, but you could have specified the amount of additional space desired:

- **-L +300M** to add the new space using size in MiB.
- **-l 175** to specify the total number of extents (175 PEs x 4 MiB).
- **-l +75** to add the additional extents needed.

8. Use **xfs_growfs** to extend the XFS file system to the remainder of the free space on the LV.

```
[root@servera ~]# xfs_growfs /data
meta-data=/dev/mapper/servera_01_vg-servera_01_lv isize=512    agcount=4,
 agsize=25600 blks
...output omitted...
```

9. Use **df** and **ls | wc** to review the new file-system size and verify that the previously existing files are still present.

```
[root@servera ~]# df -h /data
Filesystem                              Size  Used Avail Use% Mounted on
/dev/mapper/servera_01_vg-servera_01_lv  695M   26M  670M   4% /data
[root@servera ~]# ls /data | wc -l
34
```

The files still exist and the file system approximates the specified size.

10. Log off from `servera`.

```
[root@servera ~]# exit
logout
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```

Finish

On `workstation`, run the **`lab lvm-extending finish`** command to finish this exercise. This script removes the storage configured on `servera` during the exercise.

```
[student@workstation ~]$ lab lvm-extending finish
```

This concludes the guided exercise.

LAB

MANAGING LOGICAL VOLUMES

PERFORMANCE CHECKLIST

In this lab, you will resize an existing logical volume, add LVM resources as necessary, and then add a new logical volume with a persistently mounted XFS file system on it.

OUTCOMES

You should be able to:

- Resize the `serverb_01_lv` logical volume to 768 MiB.
- Create a new 128 MiB logical volume called `serverb_02_lv` with an XFS file system, persistently mounted at **`/storage/data2`**.

BEFORE YOU BEGIN

Log in to `workstation` as `student` using `student` as the password.

On `workstation`, run the **`lab lvm-review start`** command. This command runs a start script that determines if the `serverb` machine is reachable on the network. It also prepares the storage on `serverb` for the exercise.

```
[student@workstation ~]$ lab lvm-review start
```

On `serverb`, a logical volume called `serverb_01_lv` mounted at **`/storage/data1`** is running out of disk space, and you have been asked to extend it to 768 MiB in size. You must ensure that `serverb_01_lv` remains persistently mounted on **`/storage/data1`**.

You have also been asked to create a new 128 MiB logical volume called `serverb_02_lv`, mounted on **`/storage/data2`**. You have been instructed to format the new logical volume with the XFS file system.

The `serverb_01_vg` volume group holds the logical volumes. Unfortunately, it has insufficient space to extend the existing logical volume and add the new one. A 512 MiB partition was created previously on **`/dev/vdb`**. You have been instructed to use a further 512 MiB on **`/dev/vdb`**. You must create the new partition.

1. Create a 512 MiB partition on **`/dev/vdb`**, initialize it as a physical volume, and extend the `serverb_01_vg` volume group with it.
2. Extend the `serverb_01_lv` logical volume to 768 MiB, including the file system.
3. In the existing volume group, create a new logical volume called `serverb_02_lv` with a size of 128 MiB. Add an XFS file system and mount it persistently on **`/storage/data2`**.
4. When you are done, reboot your `serverb` machine, then run the command **`lab lvm-review grade`** from your `workstation` machine to verify your work.

   Wait until `serverb` is completely up and then proceed to run the evaluation.

Evaluation

On `workstation`, run the **`lab lvm-review grade`** script to confirm success on this exercise.

```
[student@workstation ~]$ lab lvm-review grade
```

Finish

On `workstation`, run the **`lab lvm-review finish`** script to complete the lab.

```
[student@workstation ~]$ lab lvm-review finish
```

This concludes the lab.

SOLUTION

MANAGING LOGICAL VOLUMES

PERFORMANCE CHECKLIST

In this lab, you will resize an existing logical volume, add LVM resources as necessary, and then add a new logical volume with a persistently mounted XFS file system on it.

OUTCOMES

You should be able to:

- Resize the `serverb_01_lv` logical volume to 768 MiB.
- Create a new 128 MiB logical volume called `serverb_02_lv` with an XFS file system, persistently mounted at **`/storage/data2`**.

BEFORE YOU BEGIN

Log in to `workstation` as `student` using `student` as the password.

On `workstation`, run the **`lab lvm-review start`** command. This command runs a start script that determines if the `serverb` machine is reachable on the network. It also prepares the storage on `serverb` for the exercise.

```
[student@workstation ~]$ lab lvm-review start
```

On `serverb`, a logical volume called `serverb_01_lv` mounted at **`/storage/data1`** is running out of disk space, and you have been asked to extend it to 768 MiB in size. You must ensure that `serverb_01_lv` remains persistently mounted on **`/storage/data1`**.

You have also been asked to create a new 128 MiB logical volume called `serverb_02_lv`, mounted on **`/storage/data2`**. You have been instructed to format the new logical volume with the XFS file system.

The `serverb_01_vg` volume group holds the logical volumes. Unfortunately, it has insufficient space to extend the existing logical volume and add the new one. A 512 MiB partition was created previously on **`/dev/vdb`**. You have been instructed to use a further 512 MiB on **`/dev/vdb`**. You must create the new partition.

1. Create a 512 MiB partition on **`/dev/vdb`**, initialize it as a physical volume, and extend the `serverb_01_vg` volume group with it.

1.1. Log in to `serverb` as the `student` user.

```
[student@workstation ~]$ ssh student@serverb
...output omitted...
[student@serverb ~]$
```

1.2. Use the **`sudo -i`** command to switch to the `root` user. The password for the `student` user is `student`.

```
[student@serverb ~]$ sudo -i
[sudo] password for student: student
[root@serverb ~]#
```

1.3. Use **parted** to create the 512 MiB partition and set it to type Linux LVM.

```
[root@serverb ~]# parted -s /dev/vdb mkpart primary 514MiB 1026MiB
[root@serverb ~]# parted -s /dev/vdb set 2 lvm on
```

1.4. Use **udevadm settle** for the system to register the new partition.

```
[root@servera ~]# udevadm settle
```

1.5. Use **pvcreate** to initialize the partition as a PV.

```
[root@serverb ~]# pvcreate /dev/vdb2
  Physical volume "/dev/vdb2" successfully created.
```

1.6. Use **vgextend** to extend the VG named `serverb_01_vg`, using the new **/dev/vdb2** PV.

```
[root@serverb ~]# vgextend serverb_01_vg /dev/vdb2
  Volume group "serverb_01_vg" successfully extended
```

2. Extend the `serverb_01_lv` logical volume to 768 MiB, including the file system.

2.1. Use **`lvextend`** to extend the `serverb_01_lv` LV to 768 MiB.

```
[root@serverb ~]# lvextend -L 768M /dev/serverb_01_vg/serverb_01_lv
  Size of logical volume serverb_01_vg/serverb_01_lv changed from
 256.00 MiB (64 extents) to 768.00 MiB (192 extents).
  Logical volume serverb_01_vg/serverb_01_lv successfully resized.
```

NOTE
Alternatively, you could have used the **`-L +512M`** option to resize the LV.

2.2. Use **`xfs_growfs`** to extend the XFS file system to the remainder of the free space on the LV.

```
[root@serverb ~]# xfs_growfs /storage/data1
meta-data=/dev/mapper/serverb_01_vg-serverb_01_lv isize=512
 agcount=4, agsize=16384 blks
...output omitted...
```

NOTE
This example shows the **`xfs_growfs`** step to extend the file system. An alternative would have been to add the **`-r`** option to the **`lvextend`** command.

3. In the existing volume group, create a new logical volume called `serverb_02_lv` with a size of 128 MiB. Add an XFS file system and mount it persistently on **`/storage/data2`**.

   3.1. Use **`lvcreate`** to create a 128 MiB LV named `serverb_02_lv` from the `serverb_01_vg` VG.

   ```
   [root@serverb ~]# lvcreate -n serverb_02_lv -L 128M serverb_01_vg
     Logical volume "serverb_02_lv" created
   ```

   3.2. Use **`mkfs`** to place an `xfs` file system on the `serverb_02_lv` LV. Use the LV device name.

   ```
   [root@serverb ~]# mkfs -t xfs /dev/serverb_01_vg/serverb_02_lv
   meta-data=/dev/serverb_01_vg/serverb_02_lv isize=512    agcount=4,
    agsize=8192 blks
   ...output omitted...
   ```

   3.3. Use **`mkdir`** to create a mount point at **`/storage/data2`**.

   ```
   [root@serverb ~]# mkdir /storage/data2
   ```

   3.4. Add the following line to the end of **`/etc/fstab`** on `serverb`:

   ```
   /dev/serverb_01_vg/serverb_02_lv  /storage/data2  xfs  defaults  1 2
   ```

   3.5. Use **`systemctl daemon-reload`** to update `systemd` with the new **`/etc/fstab`** configuration.

   ```
   [root@servera ~]# systemctl daemon-reload
   ```

   3.6. Use **`mount`** to verify the **`/etc/fstab`** entry and mount the new `serverb_02_lv` LV device.

   ```
   [root@serverb ~]# mount /storage/data2
   ```

4. When you are done, reboot your `serverb` machine, then run the command **`lab lvm-review grade`** from your `workstation` machine to verify your work.

   ```
   [root@serverb ~]# systemctl reboot
   ```

   Wait until `serverb` is completely up and then proceed to run the evaluation.

Evaluation

On `workstation`, run the **`lab lvm-review grade`** script to confirm success on this exercise.

```
[student@workstation ~]$ lab lvm-review grade
```

Finish

On `workstation`, run the **`lab lvm-review finish`** script to complete the lab.

```
[student@workstation ~]$ lab lvm-review finish
```

This concludes the lab.

SUMMARY

In this chapter, you learned:

- LVM allows you to create flexible storage by allocating space on multiple storage devices.
- Physical volumes, volume groups, and logical volumes are managed by a variety of tools such as **`pvcreate`**, **`vgreduce`**, and **`lvextend`**.
- Logical volumes can be formatted with a file system or swap space, and they can be mounted persistently.
- Additional storage can be added to volume groups and logical volumes can be extended dynamically.

CHAPTER 8

IMPLEMENTING ADVANCED STORAGE FEATURES

GOAL

Manage storage using the Stratis local storage management system and use the VDO volumes to optimize storage space in use.

OBJECTIVES

- Manage multiple storage layers using Stratis local storage management.
- Optimize the use of storage space by using VDO to compress and deduplicate data on storage devices.

SECTIONS

- Managing Layered Storage with Stratis (and Guided Exercise)
- Compressing and Deduplicating Storage with VDO (and Guided Exercise)

LAB

Implementing Advanced Storage Features

MANAGING LAYERED STORAGE WITH STRATIS

OBJECTIVES

After completing this section, you should be able to manage multiple storage layers using Stratis local storage management.

DESCRIBING THE STRATIS ARCHITECTURE

The current local storage solution in Red Hat Enterprise Linux (RHEL) includes many stable and mature technologies, including the device mapper (dm), the logical volume manager (LVM) and the XFS file system. Features provided by these components include massively scalable file systems, snapshots, redundant (RAID) logical devices, multipathing, thin provisioning, caching, deduplication, and support for virtual machines and containers. Each storage stack layer (dm, LVM, and XFS) is managed using layer-specific commands and utilities, requiring that system administrators manage physical devices, fixed-size volumes, and file systems as separate storage components.

A new generation of storage management solutions appeared in recent years, referred to as *volume-managing file systems*, that dynamically and transparently manage the volume layer as file systems are created and sized. However, although the community development of these file systems was ongoing for years, none reached the level of feature support and stability required to become the primary local storage for Red Hat Enterprise Linux.

With RHEL 8, Red Hat introduces the Stratis storage management solution. Instead of developing from scratch, as other storage projects attempted, Stratis works with existing RHEL storage components. Stratis runs as a service that manages pools of physical storage devices, and transparently creates and manages volumes for the file systems being created. Because Stratis uses existing storage drivers and tools, all of the advanced storage features that you currently use in LVM, XFS, and the device mapper are also supported by Stratis.

In a volume-managed file system, file systems are built inside shared *pools* of disk devices using a concept known as *thin provisioning*. Stratis file systems do not have fixed sizes and no longer preallocate unused block space. Although the file system is still built on a hidden LVM volume, Stratis manages the underlying volume for you and can expand it when needed. The in-use size of a file system is seen as the amount of actual blocks in use by contained files. The space available to a file system is the amount of space still unused in the pooled devices on which it resides. Multiple file systems can reside in the same pool of disk devices, sharing the available space, but file systems can also reserve pool space to guarantee availability when needed.

Stratis uses stored metadata to recognize managed pools, volumes, and file systems. Therefore, file systems created by Stratis should never be reformatted or reconfigured manually; they should only be managed using Stratis tools and commands. Manually configuring Stratis file systems could cause the loss of that metadata and prevent Stratis from recognizing the file systems it has created.

You can create multiple pools with different sets of block devices. From each pool, you can create one or more file systems. Currently, you can create up to 2^{24} file systems per pool. The following diagram illustrates how the elements of the Stratis storage management solution are positioned.

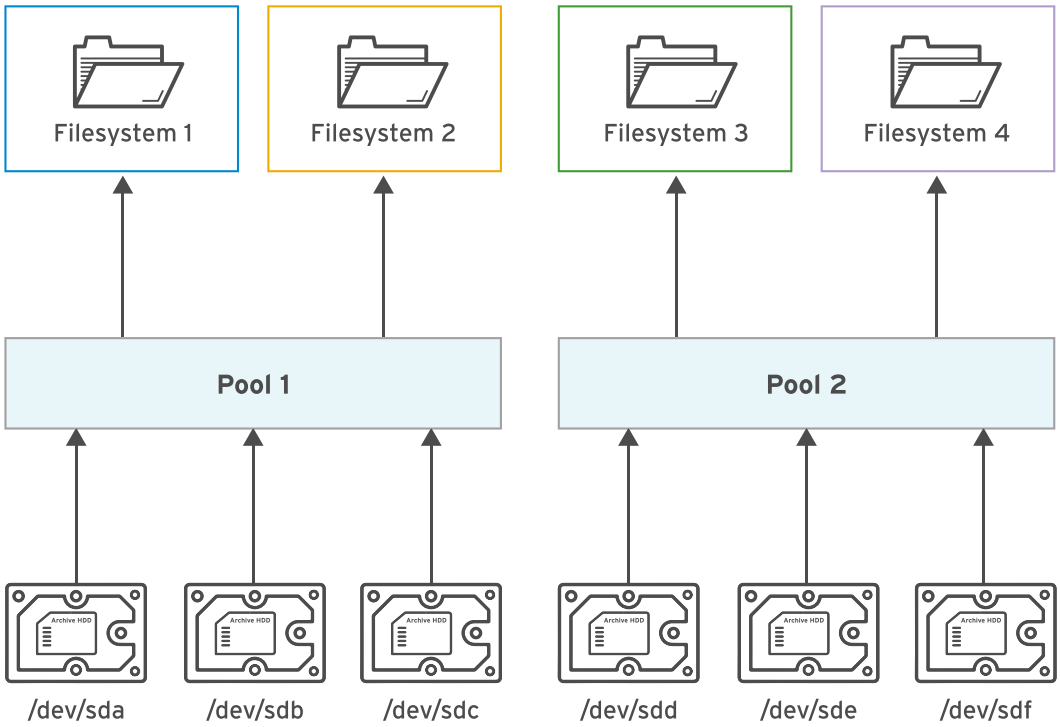

Figure 8.1: Elements of Stratis

A pool groups block devices into the *data tier* and optionally the *cache tier*. The data tier focuses on flexibility and integrity and the cache tier focuses on improved performance. Because the cache tier is intended to improve performance, you should use block devices that have higher input/output per second (IOPS), such as SSDs.

Describing the Simplified Storage Stack

Stratis simplifies many aspects of local storage provisioning and configuration across a range of Red Hat products. For example, in earlier versions of the ***Anaconda*** installer, system administrators had to layer each aspect of disk management over the other. Now, the installer uses Stratis, simplifying disk setup. Other products that use Stratis include ***Cockpit***, Red Hat Virtualization, and Red Hat Enterprise Linux Atomic Host. For all of these products, Stratis makes it simpler and less error prone to manage storage space and snapshots. Stratis allows easier integration with the higher-level management tools than using any CLI programmatically.

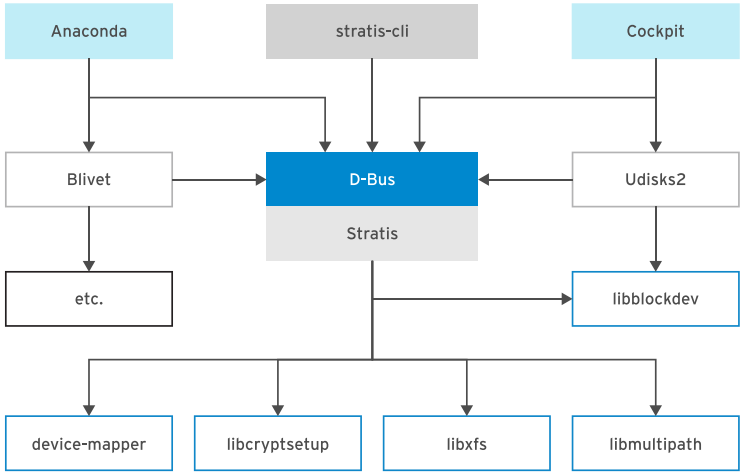

Figure 8.2: Stratis in the Linux storage management stack

Describing Stratis Layers

Internally, Stratis uses the `Backstore` subsystem to manage the block devices, and the `Thinpool` subsystem to manage the pools. The `Backstore` subsystem has a data tier that maintains the on-disk metadata on block devices, and detects and corrects data corruption. The cache tier uses high-performance block devices to act as a cache on top of the data tier. The `Thinpool` subsystem manages the thin-provisioned volumes associated with the Stratis file systems. This subsystem uses the `dm-thin` device mapper driver to replace LVM on the virtual volume sizing and management. `dm-thin` creates volumes with a large virtual size, formatted with XFS, but with a small physical size. As the physical size nears full, Stratis enlarges it automatically.

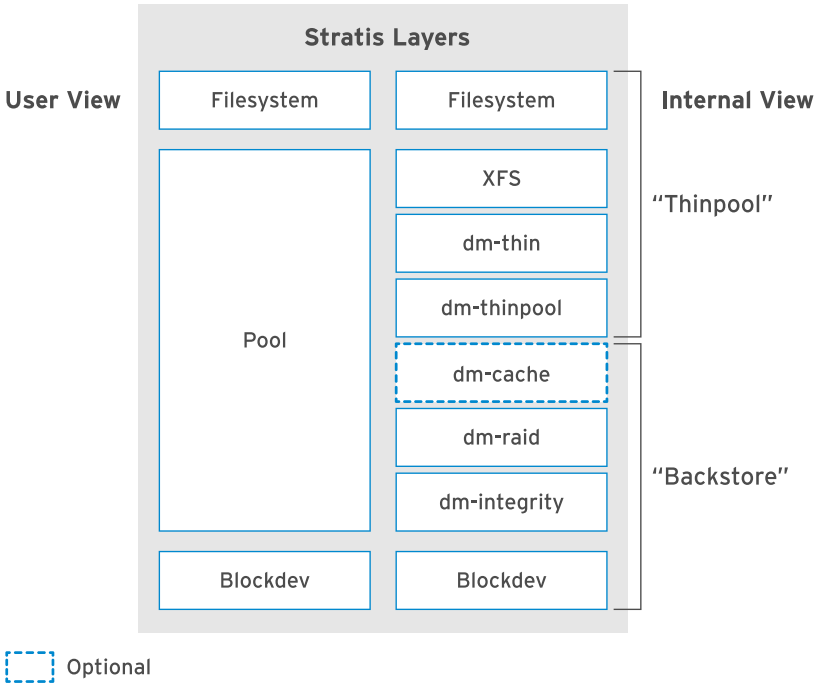

Figure 8.3: Stratis layers

Managing Thin-provisioned File Systems

To manage the thin-provisioned file systems using the Stratis storage management solution, install the *stratis-cli* and *stratisd* packages. The *stratis-cli* package provides the **`stratis`** command, which translates user requests to the `stratisd` service via the `D-Bus` API. The *stratisd* package provides the `stratisd` service, which implements the `D-Bus` interface, and manages and monitors the elements of Stratis, such as block devices, pools, and file systems. The `D-Bus` API is available if the `stratisd` service is running.

Install and activate Stratis using the usual tools:

- Install *stratis-cli* and *stratisd* using the **`yum install`** command.

```
[root@host ~]# yum install stratis-cli stratisd
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
```

- Activate the `stratisd` service using the **`systemctl`** command.

```
[root@host ~]# systemctl enable --now stratisd
```

The following are common management operations performed using the Stratis storage management solution.

- Create pools of one or more block devices using the **`stratis pool create`** command.

```
[root@host ~]# stratis pool create pool1 /dev/vdb
```

  Each pool is a subdirectory under the **`/stratis`** directory.

- Use the **`stratis pool list`** command to view the list of available pools.

```
[root@host ~]# stratis pool list
Name     Total Physical Size  Total Physical Used
pool1                  5 GiB               52 MiB
```

- Use the **`stratis pool add-data`** command to add additional block devices to a pool.

```
[root@host ~]# stratis pool add-data pool1 /dev/vdc
```

- Use the **`stratis blockdev list`** command to view the block devices of a pool.

```
[root@host ~]# stratis blockdev list pool1
Pool Name  Device Node    Physical Size   State  Tier
pool1      /dev/vdb               5 GiB  In-use  Data
pool1      /dev/vdc               5 GiB  In-use  Data
```

- Use the **`stratis filesystem create`** command to create a dynamic and flexible file system from a pool.

```
[root@host ~]# stratis filesystem create pool1 filesystem1
```

The links to the Stratis file systems are in the **/stratis/*pool1*** directory.

- Stratis supports file-system snapshotting with the **stratis filesystem snapshot** command. Snapshots are independent of the source file systems.

```
[root@host ~]# stratis filesystem snapshot pool1 filesystem1 snapshot1
```

- Use the **stratis filesystem list** command to view the list of available file systems.

```
[root@host ~]# stratis filesystem list
...output omitted...
```

To ensure that the Stratis file systems are persistently mounted, edit **/etc/fstab** and specify the details of the file system. The following command displays the UUID of the file system that you should use in **/etc/fstab** to identify the file system.

```
[root@host ~]# lsblk --output=UUID /stratis/pool1/filesystem1
UUID
31b9363b-add8-4b46-a4bf-c199cd478c55
```

The following is an example entry in the **/etc/fstab** file to persistently mount a Stratis file system.

```
UUID=31b9...8c55 /dir1 xfs defaults,x-systemd.requires=stratisd.service 0 0
```

The **x-systemd.requires=stratisd.service** mount option delays mounting the file system until after `systemd` starts the `stratisd.service` during the boot process.

NOTE

Failure to use the **x-systemd.requires=stratisd.service** mount option in **/etc/fstab** for the Stratis file system will result in the machine booting to `emergency.target` on the next reboot.

REFERENCES

For more information, refer to the *Managing layered local storage with Stratis* chapter in the *Red Hat Enterprise Linux 8 Configuring and Managing File Systems Guide* at
https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/configuring_and_managing_file_systems/

Stratis Storage
https://stratis-storage.github.io/

What Stratis learned from ZFS, Btrfs, and Linux Volume Manager
https://opensource.com/article/18/4/stratis-lessons-learned

GUIDED EXERCISE

MANAGING LAYERED STORAGE WITH STRATIS

In this exercise, you will use the Stratis storage management solution to create pools, volumes, and file systems that work in cooperation.

OUTCOMES

You should be able to:

- Create a thin-provisioned file system using Stratis storage management solution.
- Verify that the Stratis volumes grow dynamically to support real-time data growth.
- Access data from the snapshot of a thin-provisioned file system.

BEFORE YOU BEGIN

Log in to `workstation` as `student` using `student` as the password.

On `workstation`, run **`lab advstorage-stratis start`** to start the exercise. This script sets up the environment correctly and ensures that the additional disks on `servera` are clean.

```
[student@workstation ~]$ lab advstorage-stratis start
```

1. From `workstation`, open an SSH session to `servera` as `student`.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

2. Switch to the `root` user.

```
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

3. Install the *stratisd* and *stratis-cli* packages using **`yum`**.

```
[root@servera ~]# yum install stratisd stratis-cli
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
```

4. Activate the `stratisd` service using the **`systemctl`** command.

```
[root@servera ~]# systemctl enable --now stratisd
```

5. Ensure that the `stratispool1` Stratis pool exists with the block device `/dev/vdb`.

   5.1. Create a Stratis pool named `stratispool1` using the **`stratis pool create`** command.

```
[root@servera ~]# stratis pool create stratispool1 /dev/vdb
```

   5.2. Verify the availability of `stratispool1` using the **`stratis pool list`** command.

```
[root@servera ~]# stratis pool list
Name            Total Physical Size  Total Physical Used
stratispool1                  5 GiB               52 MiB
```

   Note the size of the pool in the preceding output.

6. Expand the capacity of `stratispool1` using the `/dev/vdc` block device.

   6.1. Add the block device `/dev/vdc` to `stratispool1` using the **`stratis pool add-data`** command.

```
[root@servera ~]# stratis pool add-data stratispool1 /dev/vdc
```

   6.2. Verify the size of `stratispool1` using the **`stratis pool list`** command.

```
[root@servera ~]# stratis pool list
Name            Total Physical Size  Total Physical Used
stratispool1                 10 GiB               56 MiB
```

   As shown above, the `stratispool1` pool size increased when you added the block device.

   6.3. Verify the block devices that are currently members of `stratispool1` using the **`stratis blockdev list`** command.

```
[root@servera ~]# stratis blockdev list stratispool1
Pool Name     Device Node    Physical Size   State  Tier
stratispool1  /dev/vdb               5 GiB  In-use  Data
stratispool1  /dev/vdc               5 GiB  In-use  Data
```

7. Add a thin-provisioned file system named `stratis-filesystem1` in the pool `stratispool1`. Mount the file system on **`/stratisvol`**. Create a file on the `stratis-filesystem1` file system named **`file1`** containing the text **`Hello World!`**.

7.1. Create the thin-provisioned file system `stratis-filesystem1` on `stratispool1` using the **`stratis filesystem create`** command. It may take up to a minute for the command to complete.

```
[root@servera ~]# stratis filesystem create stratispool1 stratis-
filesystem1
```

7.2. Verify the availability of `stratis-filesystem1` using the **`stratis filesystem list`** command.

```
[root@servera ~]# stratis filesystem list
Pool Name     Name                 Used     Created            Device
                                    UUID
stratispool1  stratis-filesystem1  546 MiB  Mar 29 2019 07:48  /
stratis/stratispool1/stratis-filesystem1  8714...e7db
```

Note the current usage of `stratis-filesystem1`. This usage of the file system increases on-demand in the following steps.

7.3. Create a directory named **`/stratisvol`** using the **`mkdir`** command.

```
[root@servera ~]# mkdir /stratisvol
```

7.4. Mount `stratis-filesystem1` on **`/stratisvol`** using the **`mount`** command.

```
[root@servera ~]# mount /stratis/stratispool1/stratis-filesystem1 /
stratisvol
```

7.5. Verify that the Stratis file system `stratis-filesystem1` is mounted on **`/stratisvol`** using the **`mount`** command.

```
[root@servera ~]# mount
...output omitted...
/dev/mapper/stratis-1-5c0e...12b9-thin-
fs-8714...e7db on /stratisvol type xfs
 (rw,relatime,seclabel,attr2,inode64,sunit=2048,swidth=2048,noquota)
```

7.6. Create the text file **`/stratisvol/file1`** using the **`echo`** command.

```
[root@servera ~]# echo "Hello World!" > /stratisvol/file1
```

8. Verify that the thin-provisioned file system `stratis-filesystem1` dynamically grows as the data on the file system grows.

8.1. View the current usage of `stratis-filesystem1` using the **`stratis filesystem list`** command.

```
[root@servera ~]# stratis filesystem list
Pool Name     Name                 Used     Created            Device
                                    UUID
stratispool1  stratis-filesystem1  546 MiB  Mar 29 2019 07:48  /
stratis/stratispool1/stratis-filesystem1  8714...e7db
```

8.2. Create a 2 GiB file on `stratis-filesystem1` using the **`dd`** command. It may take up to a minute for the command to complete.

```
[root@servera ~]# dd if=/dev/urandom of=/stratisvol/file2 bs=1M
 count=2048
```

8.3. Verify the usage of `stratis-filesystem1` using the **`stratis filesystem list`** command.

```
[root@servera ~]# stratis filesystem list
Pool Name     Name                 Used      Created            Device
                                     UUID
stratispool1  stratis-filesystem1  2.53 GiB  Mar 29 2019 07:48  /
stratis/stratispool1/stratis-filesystem1  8714...e7db
```

The preceding output shows that the usage of `stratis-filesystem1` has increased. The increase in the usage confirms that the thin-provisioned file system has dynamically expanded to accommodate the real-time data growth you caused by creating **`/stratisvol/file2`**.

9. Create a snapshot of `stratis-filesystem1` named `stratis-filesystem1-snap`. The snapshot will provide you with access to any file that is deleted from `stratis-filesystem1`.

   9.1. Create a snapshot of `stratis-filesystem1` using the **`stratis filesystem snapshot`** command. It may take up to a minute for the command to complete.

```
[root@servera ~]# stratis filesystem snapshot stratispool1 \
stratis-filesystem1 stratis-filesystem1-snap
```

   9.2. Verify the availability of the snapshot using the **`stratis filesystem list`** command.

```
[root@servera ~]# stratis filesystem list
...output omitted...
stratispool1  stratis-filesystem1-snap  2.53 GiB  Mar 29 2019 10:28  /
stratis/stratispool1/stratis-filesystem1-snap  291d...8a16
```

   9.3. Remove the **`/stratisvol/file1`** file.

```
[root@servera ~]# rm /stratisvol/file1
rm: remove regular file '/stratisvol/file1'? y
```

   9.4. Create the directory **`/stratisvol-snap`** using the **`mkdir`** command.

```
[root@servera ~]# mkdir /stratisvol-snap
```

   9.5. Mount the `stratis-filesystem1-snap` snapshot on **`/stratisvol-snap`** using the **`mount`** command.

```
[root@servera ~]# mount /stratis/stratispool1/stratis-filesystem1-snap
 \
/stratisvol-snap
```

   9.6. Confirm that you can still access the file you deleted from `stratis-filesystem1` using the `stratis-filesystem1-snap` snapshot.

```
[root@servera ~]# cat /stratisvol-snap/file1
Hello World!
```

10. Unmount **`/stratisvol`** and **`/stratisvol-snap`** using the **`umount`** command.

```
[root@servera ~]# umount /stratisvol-snap
[root@servera ~]# umount /stratisvol
```

11. Remove the thin-provisioned file system `stratis-filesystem1` and its snapshot `stratis-filesystem1-snap` from the system.

    11.1. Destroy `stratis-filesystem1-snap` using the **`stratis filesystem destroy`** command.

    ```
    [root@servera ~]# stratis filesystem destroy stratispool1 stratis-
    filesystem1-snap
    ```

    11.2. Destroy `stratis-filesystem1` using the **`stratis filesystem destroy`** command.

    ```
    [root@servera ~]# stratis filesystem destroy stratispool1 stratis-
    filesystem1
    ```

    11.3. Exit the `root` user's shell.

    ```
    [root@servera ~]# exit
    logout
    [student@servera ~]$
    ```

    11.4. Log off from `servera`.

    ```
    [student@servera ~]$ exit
    logout
    Connection to servera closed.
    [student@workstation ~]$
    ```

Finish

On `workstation`, run **`lab advstorage-stratis finish`** to complete this exercise. This script deletes the partitions and files created throughout the exercise and ensures that the environment is clean.

```
[student@workstation ~]$ lab advstorage-stratis finish
```

This concludes the guided exercise.

COMPRESSING AND DEDUPLICATING STORAGE WITH VDO

OBJECTIVES

After completing this section, you should be able to optimize the use of storage space by using VDO to compress and deduplicate data on storage devices.

DESCRIBING VIRTUAL DATA OPTIMIZER

Red Hat Enterprise Linux 8 includes the Virtual Data Optimizer (VDO) driver, which optimizes the data footprint on block devices. VDO is a Linux device mapper driver that reduces disk space usage on block devices, and minimizes the replication of data, saving disk space and even increasing data throughput. VDO includes two kernel modules: the `kvdo` module to transparently control data compression, and the `uds` module for deduplication.

The VDO layer is placed on top of an existing block storage device, such as a RAID device or a local disk. Those block devices can also be encrypted devices. The storage layers, such as LVM logical volumes and file systems, are placed on top of a VDO device. The following diagram shows the placement of VDO in an infrastructure consisting of KVM virtual machines that are using optimized storage devices.

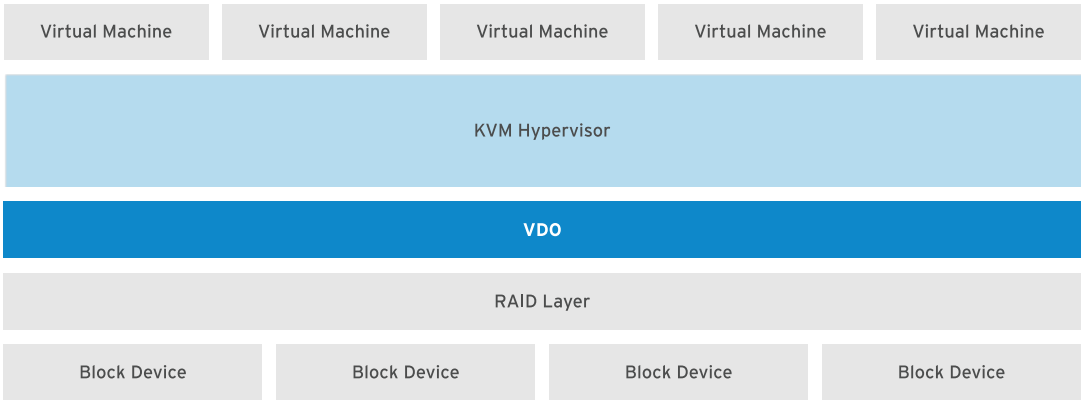

Figure 8.4: VDO-based virtual machines

VDO applies three phases to data in the following order to reduce the footprint on storage devices:

1. *Zero-Block Elimination* filters out data blocks that contain only zeroes (0) and records the information of those blocks only in the metadata. The nonzero data blocks are then passed to the next phase of processing. This phase enables the thin provisioning feature in the VDO devices.

2. *Deduplication* eliminates redundant data blocks. When you create multiple copies of the same data, VDO detects the duplicate data blocks and updates the metadata to use those duplicate blocks as references to the original data block without creating redundant data blocks. The universal deduplication service (UDS) kernel module checks redundancy of the data through the metadata it maintains. This kernel module ships as part of the VDO.

3. *Compression* is the last phase. The `kvdo` kernel module compresses the data blocks using LZ4 compression and groups them on 4 KB blocks.

IMPLEMENTING VIRTUAL DATA OPTIMIZER

The logical devices that you create using VDO are called *VDO volumes*. VDO volumes are similar to disk partitions; you can format the volumes with the desired file-system type and mount it like a regular file system. You can also use a VDO volume as an LVM physical volume.

To create a VDO volume, specify a block device and the name of the logical device that VDO presents to the user. You can optionally specify the logical size of the VDO volume. The logical size of the VDO volume can be more than the physical size of the actual block device.

Because the VDO volumes are thinly provisioned, users can only see the logical space in use and are unaware of the actual physical space available. If you do not specify the logical size while creating the volume, VDO assumes the actual physical size as the logical size of the volume. This 1:1 ratio of mapping logical size to physical size gives better performance but provides less efficient use of storage space. Based on your infrastructure requirements, you should prioritize either performance or space efficiency.

When the logical size of a VDO volume is more than the actual physical size, you should proactively monitor the volume statistics to view the actual usage using the **`vdostats --verbose`** command.

Enabling VDO

Install the *vdo* and *kmod-kvdo* packages to enable VDO in the system.

```
[root@host ~]# yum install vdo kmod-kvdo
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
```

Creating a VDO Volume

To create a VDO volume, run the **`vdo create`** command.

```
[root@host ~]# vdo create --name=vdo1 --device=/dev/vdd --vdoLogicalSize=50G
...output omitted...
```

If you omit the logical size, the resulting VDO volume gets the same size as its physical device.

When the VDO volume is in place, you can format it with the file-system type of your choice and mount it under the file-system hierarchy on your system.

Analyzing a VDO Volume

To analyze a VDO volume, run the **`vdo status`** command. This command displays a report on the VDO system, and the status of the VDO volume in YAML format. It also displays attributes of the VDO volume. Use the **`--name=`** option to specify the name of a particular volume. If you omit the name of the specific volume, the output of the **`vdo status`** command displays the status of all the VDO volumes.

```
[root@host ~]# vdo status --name=vdo1
...output omitted...
```

The **`vdo list`** command displays the list of VDO volumes that are currently started. You can start and stop a VDO volume using the **`vdo start`** and **`vdo stop`** commands, respectively.

REFERENCES

For more information, refer to the *Getting started with VDO* chapter in the *Red Hat Enterprise Linux 8 Deduplicating and Compressing Storage Guide* at https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/deduplicating_and_compressing_storage/

Introducing Virtual Data Optimizer
https://rhelblog.redhat.com/2018/04/11/introducing-virtual-data-optimizer-to-reduce-cloud-and-on-premise-storage-costs/

GUIDED EXERCISE

COMPRESSING AND DEDUPLICATING STORAGE WITH VDO

In this exercise, you will create a VDO volume, format it with a file system, mount it, store data on it, and investigate the impact of compression and deduplication on storage space actually used.

OUTCOMES

You should be able to:

- Create a volume using Virtual Data Optimizer, format it with a file-system type, and mount a file system on it.
- Investigate the impact of data deduplication and compression on a Virtual Data Optimizer volume.

BEFORE YOU BEGIN

Log in to `workstation` as `student` using `student` as the password.

On `workstation`, run **`lab advstorage-vdo start`** to start the exercise. This script ensures that there are no partitions on the `/dev/vdd` disk and sets up the environment correctly.

```
[student@workstation ~]$ lab advstorage-vdo start
```

1. From `workstation`, open an SSH session to `servera` as `student`.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

2. Create the VDO volume `vdo1`, using the `/dev/vdd` device. Set its logical size to 50 GB.

   2.1. Switch to the `root` user.

```
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

   2.2. Confirm that the *vdo* package is installed using the **`rpm`** command.

```
[root@servera ~]# yum list installed vdo
vdo-6.2.0.293-10.el8.x86_64
```

2.3. Create the `vdo1` volume using the **`vdo create`** command.

```
[root@servera ~]# vdo create --name=vdo1 --device=/dev/vdd --
vdoLogicalSize=50G
...output omitted...
```

2.4. Verify the availability of the `vdo1` volume using the **`vdo list`** command.

```
[root@servera ~]# vdo list
vdo1
```

3. Verify that the `vdo1` volume has both the compression and deduplication features enabled.

3.1. Use **`grep`** to search for the lines containing the string **`Deduplication`** in the output of the **`vdo status --name=vdo1`** command.

```
[root@servera ~]# vdo status --name=vdo1 | grep Deduplication
    Deduplication: enabled
```

3.2. Use **`grep`** to search for the lines containing the string **`Compression`** in the output of the **`vdo status --name=vdo1`** command.

```
[root@servera ~]# vdo status --name=vdo1 | grep Compression
    Compression: enabled
```

4. Format the `vdo1` volume with the XFS file-system type and mount it on **`/mnt/vdo1`**.

   4.1. Format the `vdo1` volume with the XFS file system using the **`mkfs`** command.

```
[root@servera ~]# mkfs.xfs -K /dev/mapper/vdo1
...output omitted...
```

   The **`-K`** option in the preceding **`mkfs.xfs`** command prevents the unused blocks in the file system from being discarded immediately which lets the command return faster.

   4.2. Use the **`udevadm`** command to register the new device node.

```
[root@servera ~]# udevadm settle
```

   4.3. Create the **`/mnt/vdo1`** directory using the **`mkdir`** command.

```
[root@servera ~]# mkdir /mnt/vdo1
```

   4.4. Mount the `vdo1` volume on **`/mnt/vdo1`** using the **`mount`** command.

```
[root@servera ~]# mount /dev/mapper/vdo1 /mnt/vdo1
```

   4.5. Verify that the `vdo1` volume is successfully mounted using the **`mount`** command.

```
[root@servera ~]# mount
...output omitted...
/dev/mapper/vdo1 on /mnt/vdo1 type xfs
 (rw,relatime,seclabel,attr2,inode64,noquota)
```

5. Create three copies of the same file named **`/root/install.img`** on the `vdo1` volume. Compare the statistics of the volume to verify the data deduplication and compression happening on the volume. The preceding output may vary on your system..

   5.1. View the initial statistics and status of the volume using the **`vdostats`** command.

```
[root@servera ~]# vdostats --human-readable
Device                    Size      Used Available Use% Space saving%
/dev/mapper/vdo1          5.0G      3.0G      2.0G  60%           99%
```

   Notice that 3 GB of the volume is already used because when created, the VDO volume reserves 3-4 GB for itself. Also, note that the value **`99%`** in the **`Space`**

saving% field indicates that you have not created any content so far in the volume contributing to all of the saved volume space.

5.2. Copy **/root/install.img** to **/mnt/vdo1/install.img.1** and verify the statistics of the volume. It may take up to a minute to copy the file.

```
[root@servera ~]# cp /root/install.img /mnt/vdo1/install.img.1
[root@servera ~]# vdostats --human-readable
Device                    Size      Used Available Use% Space saving%
/dev/mapper/vdo1          5.0G      3.4G      1.6G  68%            5%
```

Notice that the value of the **Used** field increased from **3.0G** to **3.4G** because you copied a file to the volume, and that occupies some space. Also, notice that the value of **Space saving%** field decreased from **99%** to **5%** because initially there was no content in the volume, contributing to the low volume space utilization and high volume space saving until you created a file. The volume space saving is considerably low because you created a unique copy of the file in the volume and there is nothing to deduplicate.

5.3. Copy **/root/install.img** to **/mnt/vdo1/install.img.2** and verify the statistics of the volume. It may take up to a minute to copy the file.

```
[root@servera ~]# cp /root/install.img /mnt/vdo1/install.img.2
[root@servera ~]# vdostats --human-readable
Device                    Size      Used Available Use% Space saving%
/dev/mapper/vdo1          5.0G      3.4G      1.6G  68%           51%
```

Notice that the used volume space did not change rather the percentage of the saved volume space increased proving that the data deduplication occurred to reduce the space consumption for the redundant copies of the same file. The value of **Space saving%** in the preceding output may vary on your system.

5.4. Exit the `root` user's shell.

```
[root@servera ~]# exit
logout
[student@servera ~]$
```

5.5. Log off from `servera`.

```
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```

Finish

On `workstation`, run **lab advstorage-vdo finish** to complete this exercise. This script deletes the files created throughout the exercise and ensures that the environment is clean.

```
[student@workstation ~]$ lab advstorage-vdo finish
```

This concludes the guided exercise.

LAB

IMPLEMENTING ADVANCED STORAGE FEATURES

In this exercise, you will use the Stratis storage management solution to create file systems that grow to accommodate increased data demands, and Virtual Data Optimizer to create volumes for efficient utilization of storage space.

OUTCOMES

You should be able to:

- Create a thinly provisioned file system using Stratis storage management solution.
- Verify that the Stratis volumes grow dynamically to support real-time data growth.
- Access data from the snapshot of a thinly provisioned file system.
- Create a volume using Virtual Data Optimizer and mount it on a file system.
- Investigate the impact of data deduplication and compression on a Virtual Data Optimizer volume.

BEFORE YOU BEGIN

Log in to `workstation` as `student` using `student` as the password.

On `workstation`, run **`lab advstorage-review start`** to start the lab. This script sets up the environment correctly and ensures that the additional disks on `serverb` are clean.

```
[student@workstation ~]$ lab advstorage-review start
```

1. From `workstation`, open an SSH session to `serverb` as `student`.
2. Switch to the `root` user.
3. Install the *stratisd* and *stratis-cli* packages using **`yum`**.
4. Start and enable the `stratisd` service using the **`systemctl`** command.
5. Create the Stratis pool `labpool` containing the block device `/dev/vdb`.
6. Expand the capacity of `labpool` using the disk `/dev/vdc` available in the system.
7. Create a thinly provisioned file system named `labfs` in the `labpool` pool. Mount this file system on **`/labstratisvol`** so that it persists across reboots. Create a file named **`labfile1`** that contains the text **`Hello World!`** on the `labfs` file system. Don't forget to use the `x-systemd.requires=stratisd.service` mount option in **`/etc/fstab`**.
8. Verify that the thinly provisioned file system `labfs` dynamically grows as the data on the file system grows.
9. Create a snapshot named `labfs-snap` of the `labfs` file system. The snapshot allows you to access any file that is deleted from `labfs`.

10. Create the VDO volume `labvdo`, with the device `/dev/vdd`. Set its logical size to **`50 GB`**.
11. Mount the volume `labvdo` on **`/labvdovol`** with the XFS file system so that it persists across reboots. Don't forget to use the `x-systemd.requires=vdo.service` mount option in **`/etc/fstab`**.
12. Create three copies of the file named **`/root/install.img`** on the volume `labvdo`. Compare the statistics of the volume to verify the data deduplication and compression happening on the volume.

Evaluation

On `workstation`, run the **`lab advstorage-review grade`** command to confirm success of this exercise.

```
[student@workstation ~]$ lab advstorage-review grade
```

Finish

On `workstation`, run **`lab advstorage-review finish`** to complete this exercise. This script deletes the partitions and files created throughout the exercise and ensures that the environment is clean.

```
[student@workstation ~]$ lab advstorage-review finish
```

This concludes the lab.

SOLUTION

IMPLEMENTING ADVANCED STORAGE FEATURES

In this exercise, you will use the Stratis storage management solution to create file systems that grow to accommodate increased data demands, and Virtual Data Optimizer to create volumes for efficient utilization of storage space.

OUTCOMES

You should be able to:

- Create a thinly provisioned file system using Stratis storage management solution.
- Verify that the Stratis volumes grow dynamically to support real-time data growth.
- Access data from the snapshot of a thinly provisioned file system.
- Create a volume using Virtual Data Optimizer and mount it on a file system.
- Investigate the impact of data deduplication and compression on a Virtual Data Optimizer volume.

BEFORE YOU BEGIN

Log in to `workstation` as `student` using `student` as the password.

On `workstation`, run **`lab advstorage-review start`** to start the lab. This script sets up the environment correctly and ensures that the additional disks on `serverb` are clean.

```
[student@workstation ~]$ lab advstorage-review start
```

1. From `workstation`, open an SSH session to `serverb` as `student`.

```
[student@workstation ~]$ ssh student@serverb
...output omitted...
[student@serverb ~]$
```

2. Switch to the `root` user.

```
[student@serverb ~]$ sudo -i
[sudo] password for student: student
[root@serverb ~]#
```

3. Install the *stratisd* and *stratis-cli* packages using **`yum`**.

```
[root@serverb ~]# yum install stratisd stratis-cli
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
```

4. Start and enable the `stratisd` service using the **`systemctl`** command.

```
[root@serverb ~]# systemctl enable --now stratisd
```

5. Create the Stratis pool `labpool` containing the block device `/dev/vdb`.

   5.1. Create the Stratis pool `labpool` using the **`stratis pool create`** command.

```
[root@serverb ~]# stratis pool create labpool /dev/vdb
```

   5.2. Verify the availability of `labpool` using the **`stratis pool list`** command.

```
[root@serverb ~]# stratis pool list
Name            Total Physical Size  Total Physical Used
labpool                  5 GiB               52 MiB
```

   Note the size of the pool in the preceding output.

6. Expand the capacity of `labpool` using the disk `/dev/vdc` available in the system.

   6.1. Add the block device `/dev/vdc` to `labpool` using the **`stratis pool add-data`** command.

```
[root@serverb ~]# stratis pool add-data labpool /dev/vdc
```

   6.2. Verify the size of `labpool` using the **`stratis pool list`** command.

```
[root@serverb ~]# stratis pool list
Name            Total Physical Size  Total Physical Used
labpool                 10 GiB               56 MiB
```

   The preceding output shows that the size of `labpool` has increased after a new disk was added to the pool.

   6.3. Use the **`stratis blockdev list`** command to list the block devices that are now members of `labpool`.

```
[root@serverb ~]# stratis blockdev list labpool
Pool Name     Device Node    Physical Size   State  Tier
labpool  /dev/vdb               5 GiB  In-use  Data
labpool  /dev/vdc               5 GiB  In-use  Data
```

7. Create a thinly provisioned file system named `labfs` in the `labpool` pool. Mount this file system on **`/labstratisvol`** so that it persists across reboots. Create a file named

labfile1 that contains the text **Hello World!** on the labfs file system. Don't forget to use the x-systemd.requires=stratisd.service mount option in **/etc/fstab**.

7.1. Create the thinly provisioned file system labfs in labpool using the **stratis filesystem create** command. It may take up to a minute for the command to complete.

```
[root@serverb ~]# stratis filesystem create labpool labfs
```

7.2. Verify the availability of labfs using the **stratis filesystem list** command.

```
[root@serverb ~]# stratis filesystem list
Pool Name     Name                 Used     Created            Device
                                   UUID
labpool  labfs  546 MiB  Mar 29 2019 07:48  /stratis/labpool/labfs
 9825...d6ca
```

Note the current usage of labfs. This usage of the file system increases on-demand in the following steps.

7.3. Determine the UUID of labfs using the **lsblk** command.

```
[root@serverb ~]# lsblk --output=UUID /stratis/labpool/labfs
UUID
9825e289-fb08-4852-8290-44d1b8f0d6ca
```

7.4. Edit **/etc/fstab** so that the thinly provisioned file system labfs is mounted at boot time. Use the UUID you determined in the preceding step. The following shows the line you should add to **/etc/fstab**. You can use the **vi /etc/fstab** command to edit the file.

```
UUID=9825...d6ca /labstratisvol  xfs defaults,x-
systemd.requires=stratisd.service 0 0
```

7.5. Create a directory named **/labstratisvol** using the **mkdir** command.

```
[root@serverb ~]# mkdir /labstratisvol
```

7.6. Mount the thinly provisioned file system labfs using the **mount** command to confirm that the **/etc/fstab** file contains the appropriate entries.

```
[root@serverb ~]# mount /labstratisvol
```

If the preceding command produces any errors, revisit the **/etc/fstab** file and ensure that it contains the appropriate entries.

7.7. Create a text file named **/labstratisvol/labfile1** using the **echo** command.

```
[root@serverb ~]# echo "Hello World!" > /labstratisvol/labfile1
```

8. Verify that the thinly provisioned file system `labfs` dynamically grows as the data on the file system grows.

8.1. View the current usage of `labfs` using the **`stratis filesystem list`** command.

```
[root@serverb ~]# stratis filesystem list
Pool Name     Name                 Used     Created            Device
                                   UUID
labpool  labfs  546 MiB  Mar 29 2019 07:48  /stratis/labpool/labfs
 9825...d6ca
```

8.2. Create a 2 GiB file in `labfs` using the **dd** command. It may take up to a minute for the command to complete.

```
[root@serverb ~]# dd if=/dev/urandom of=/labstratisvol/labfile2 bs=1M
 count=2048
```

8.3. Verify that the usage of `labfs` has increased, using the **`stratis filesystem list`** command.

```
[root@serverb ~]# stratis filesystem list
Pool Name     Name                 Used      Created            Device
                                    UUID
labpool  labfs  2.53 GiB  Mar 29 2019 07:48  /stratis/labpool/labfs
 9825...d6ca
```

9. Create a snapshot named `labfs-snap` of the `labfs` file system. The snapshot allows you to access any file that is deleted from `labfs`.

9.1. Create a snapshot of `labfs` using the **`stratis filesystem snapshot`** command. It may take up to a minute for the command to complete.

```
[root@serverb ~]# stratis filesystem snapshot labpool \
labfs labfs-snap
```

9.2. Verify the availability of the snapshot using the **`stratis filesystem list`** command.

```
[root@serverb ~]# stratis filesystem list
...output omitted...
labpool  labfs-snap  2.53 GiB  Mar 29 2019 10:28  /stratis/labpool/
labfs-snap  291d...8a16
```

9.3. Remove the **`/labstratisvol/labfile1`** file.

```
[root@serverb ~]# rm /labstratisvol/labfile1
rm: remove regular file '/labstratisvol/labfile1'? y
```

9.4. Create the **`/labstratisvol-snap`** directory using the **`mkdir`** command.

```
[root@serverb ~]# mkdir /labstratisvol-snap
```

9.5. Mount the snapshot `labfs-snap` on **`/labstratisvol-snap`** using the **`mount`** command.

```
[root@serverb ~]# mount /stratis/labpool/labfs-snap \
/labstratisvol-snap
```

9.6. Confirm that you can still access the file you deleted from `labfs` using the snapshot `labfs-snap`.

```
[root@serverb ~]# cat /labstratisvol-snap/labfile1
Hello World!
```

10. Create the VDO volume `labvdo`, with the device `/dev/vdd`. Set its logical size to **`50 GB`**.

10.1. Create the volume `labvdo` using the **`vdo create`** command.

```
[root@serverb ~]# vdo create --name=labvdo --device=/dev/vdd --
vdoLogicalSize=50G
...output omitted...
```

10.2. Verify the availability of the volume `labvdo` using the **`vdo list`** command.

```
[root@serverb ~]# vdo list
labvdo
```

11. Mount the volume `labvdo` on **`/labvdovol`** with the XFS file system so that it persists across reboots. Don't forget to use the `x-systemd.requires=vdo.service` mount option in **`/etc/fstab`**.

11.1. Format the `labvdo` volume with the XFS file system using the **`mkfs`** command.

```
[root@serverb ~]# mkfs.xfs -K /dev/mapper/labvdo
...output omitted...
```

11.2. Use the **`udevadm`** command to register the new device node.

```
[root@serverb ~]# udevadm settle
```

11.3. Create the **`/labvdovol`** directory using the **`mkdir`** command.

```
[root@serverb ~]# mkdir /labvdovol
```

11.4. Determine the UUID of `labvdo` using the **`lsblk`** command.

```
[root@serverb ~]# lsblk --output=UUID /dev/mapper/labvdo
UUID
ef8cce71-228a-478d-883d-5732176b39b1
```

11.5. Edit **`/etc/fstab`** so that `labvdo` is mounted at boot time. Use the UUID of the volume you determined in the preceding step. The following shows the line you should add to **`/etc/fstab`**. You can use the **`vi /etc/fstab`** command to edit the file.

```
UUID=ef8c...39b1 /labvdovol xfs defaults,x-systemd.requires=vdo.service
 0 0
```

11.6. Mount the `labvdo` volume using the **`mount`** command to confirm that the **`/etc/fstab`** file contains the appropriate entries.

```
[root@serverb ~]# mount /labvdovol
```

If the preceding command produces any errors, revisit the **`/etc/fstab`** file and ensure that it contains the appropriate entries.

12. Create three copies of the file named **`/root/install.img`** on the volume `labvdo`. Compare the statistics of the volume to verify the data deduplication and compression happening on the volume.

12.1. View the initial statistics and status of the volume using the **`vdostats`** command.

```
[root@serverb ~]# vdostats --human-readable
Device                    Size      Used Available Use% Space saving%
/dev/mapper/labvdo          5.0G      3.0G      2.0G  60%           99%
```

Notice that 3 GB of the volume is already used because when created, the VDO volume reserves 3-4 GB for itself. Also note that the value **`99%`** in the **`Space`**

saving% field indicates that you have not created any content so far in the volume, contributing to all of the saved volume space.

12.2. Copy **/root/install.img** to **/labvdovol/install.img.1** and verify the statistics of the volume. It may take up to a minute to copy the file.

```
[root@serverb ~]# cp /root/install.img /labvdovol/install.img.1
[root@serverb ~]# vdostats --human-readable
Device                    Size      Used Available Use% Space saving%
/dev/mapper/labvdo          5.0G      3.4G      1.6G  68%           5%
```

Notice that the value of the **Used** field increased from **3.0G** to **3.4G** because you copied a file in the volume, and that occupies some space. Also, notice that the value of **Space saving%** field decreased from **99%** to **5%** because initially there was no content in the volume, contributing to the low volume space utilization and high volume space saving until you created a file in there. The volume space saving is quite low because you created a unique copy of the file in the volume and there is nothing to deduplicate.

12.3. Copy **/root/install.img** to **/labvdovol/install.img.2** and verify the statistics of the volume. It may take up to a minute to copy the file.

```
[root@serverb ~]# cp /root/install.img /labvdovol/install.img.2
[root@serverb ~]# vdostats --human-readable
Device                    Size      Used Available Use% Space saving%
/dev/mapper/labvdo          5.0G      3.4G      1.6G  68%          51%
```

Notice that the used volume space did not change. Instead, the percentage of the saved volume space increased, proving that the data deduplication occurred to reduce the space consumption for the redundant copies of the same file. The value of **Space saving%** in the preceding output may vary on your system.

12.4. Reboot the serverb machine.

```
[root@serverb ~]# systemctl reboot
```

NOTE

Note: If on a reboot, serverb doesn't boot to a regular login prompt but instead has "Give root password for maintenance (or press **Control-D** to continue):" you likely made a mistake in **/etc/fstab**. After providing the root password of **redhat**, you will need to remount the root file system as read-write with:

```
[root@serverb ~]# mount -o remount,rw /
```

Verify that **/etc/fstab** is configured correctly as specified in the solutions. Pay special attention to the mount options for the lines related to **/labstratisvol** and **/labvdovol**.

Evaluation

On `workstation`, run the **lab advstorage-review grade** command to confirm success of this exercise.

```
[student@workstation ~]$ lab advstorage-review grade
```

Finish

On `workstation`, run **`lab advstorage-review finish`** to complete this exercise. This script deletes the partitions and files created throughout the exercise and ensures that the environment is clean.

```
[student@workstation ~]$ lab advstorage-review finish
```

This concludes the lab.

SUMMARY

In this chapter, you learned:

- The Stratis storage management solution implements flexible file systems that grow dynamically with data.
- The Stratis storage management solution supports thin provisioning, snapshotting, and monitoring.
- The Virtual Data Optimizer (VDO) aims to reduce the cost of data storage.
- The Virtual Data Optimizer applies zero-block elimination, data deduplication, and data compression to optimize disk space efficiency.

CHAPTER 9

ACCESSING NETWORK-ATTACHED STORAGE

GOAL

Access network-attached storage using the NFS protocol.

OBJECTIVES

- Mount, use, and unmount an NFS export from the command line and at boot time.
- Configure the automounter with direct and indirect maps to automatically mount an NFS file system on demand, and unmount it when it is no longer in use.
- Configure an NFS client to use NFSv4 using the new **`nfsconf`** tool.

SECTIONS

- Mounting Network-Attached Storage with NFS (and Guided Exercise)
- Automounting Network-Attached Storage (and Guided Exercise)

LAB

Accessing Network-Attached Storage

MOUNTING NETWORK-ATTACHED STORAGE WITH NFS

OBJECTIVES

After completing this section, you should be able to:

- Identify NFS share information.
- Create a directory to use as a mount point.
- Mount an NFS share using the **`mount`** command or by configuring the **`/etc/fstab`** file.
- Unmount an NFS share using the **`umount`** command.
- Configure an NFS client to use NFSv4 using the new **`nfsconf`** tool.

MOUNTING AND UNMOUNTING NFS SHARES

NFS, the *Network File System*, is an internet standard protocol used by Linux, UNIX, and similar operating systems as their native network file system. It is an open standard, still being actively enhanced, which supports native Linux permissions and file-system features.

The default NFS version in Red Hat Enterprise Linux 8 is 4.2. NFSv4 and NFSv3 major versions are supported. NFSv2 is no longer supported. NFSv4 uses only the TCP protocol to communicate with the server; earlier NFS versions could use either TCP or UDP.

NFS servers *export* shares (directories). NFS clients mount an exported share to a local mount point (directory), which must exist. NFS shares can be mounted a number of ways:

- Manually, using the **`mount`** command.
- Automatically at boot time using **`/etc/fstab`** entries.
- On demand, using either the `autofs` service or the `systemd.automount` facility.

Mounting NFS Shares

To mount an NFS share, follow these three steps:

1. **Identify:** The administrator of the NFS client system can identify available NFS shares in various ways:

   The administrator for the NFS server may provide export details, including security requirements.

   Alternatively, the client administrator can identify NFSv4 shares by mounting the root directory of the NFS server and exploring the exported directories. Do this as the `root` user. Access to shares that use Kerberos security will be denied, but the share (directory) name will be visible. Other shared directories will be browsable.

```
[user@host ~]$ sudo mkdir mountpoint
[user@host ~]$ sudo mount serverb:/ mountpoint
[user@host ~]$ sudo ls mountpoint
```

2. **Mount point:** Use **mkdir** to create a mount point in a suitable location.

```
[user@host ~]$ mkdir -p mountpoint
```

3. **Mount:** As with file systems on partitions, NFS shares must be mounted to be available. To mount an NFS share, select from the following. In each case, you must run these commands as a superuser either by logging in as `root` or by using the **sudo** command.

   - Mount temporarily: Mount the NFS share using the **mount** command:

```
[user@host ~]$ sudo mount -t nfs -o rw,sync serverb:/share mountpoint
```

     The **-t nfs** option is the file-system type for NFS shares (not strictly required but shown for completeness). The **-o sync** option tells **mount** to immediately synchronize write operations with the NFS server (the default is asynchronous).

     This command mounts the share immediately but not persistently; the next time the system boots, this NFS share will not be available. This is useful for one-time access to data. It is also useful for test mounting an NFS share before making the share available persistently.

   - Mount persistently: To ensure that the NFS share is mounted at boot time, edit the **/etc/fstab** file to add the mount entry.

```
[user@host ~]$ sudo vim /etc/fstab
...
serverb:/share  /mountpoint  nfs  rw,soft  0 0
```

     Then, mount the NFS share:

```
[user@host ~]$ sudo mount /mountpoint
```

     Because the NFS server and mount options are found in the **/etc/fstab** file by the NFS client service, you do not need to specify these on the command line.

Unmounting NFS Shares

As `root` (or using **sudo**), unmount an NFS share using the **umount** command.

```
[user@host ~]$ sudo umount mountpoint
```

> **NOTE**
> Unmounting a share does not remove its **/etc/fstab** entry. Unless you remove or comment out the entry, the NFS share will be remounted either at the next system boot or when the NFS client service is restarted.

THE nfsconf TOOL

Red Hat Enterprise Linux 8 introduces the **nfsconf** tool to manage the NFS client and server configuration files under NFSv4 and NFSv3. Configure the **nfsconf** tool using **/etc/nfs.conf**

(the **/etc/sysconfig/nfs** file from earlier versions of the operating system is deprecated now). Use the **nfsconf** tool to get, set, or unset NFS configuration parameters.

The **/etc/nfs.conf** configuration file is composed of multiple sections starting with a key word in square brackets (***[keyword]***) with value assignments within the section. For an NFS server, configure the **[nfsd]** section. A value assignment or key is composed of a name for the value, an equals sign, and a setting for the value, such as **vers4.2=y**. The lines starting with "#" or ";" are ignored, as are any blank lines.

```
[user@host ~]$ sudo cat /etc/nfs.conf
...output omitted...
[nfsd]
# debug=0
# threads=8
# host=
# port=0
# grace-time=90
# lease-time=90
# tcp=y
# vers2=n
# vers3=y
# vers4=y
# vers4.0=y
# vers4.1=y
# vers4.2=y
# rdma=n
#
```

By default the **[nfsd]** section's key-value pairs are commented out. However, the comments show the default options that will take effect if they are unchanged. This provides you with a good starting point for NFS configuration.

Use the **nfsconf --set *section key value*** to set a value for the key in the specified section.

```
[user@host ~]$ sudo nfsconf --set nfsd vers4.2 y
```

This command updates the **/etc/nfs.conf** configuration file:

```
[user@host ~]$ sudo cat /etc/nfs.conf
...output omitted...
[nfsd]
vers4.2 = y
# debug=0
# threads=8
# host=
# port=0
# grace-time=90
# lease-time=90
# tcp=y
# vers2=n
# vers3=y
# vers4=y
```

```
# vers4.0=y
# vers4.1=y
# vers4.2=y
# rdma=n
#
```

Use the **`nfsconf --get section key`** to retrieve the value for the key in the specified section:

```
[user@host ~]$ sudo nfsconf --get nfsd vers4.2
y
```

Use the **`nfsconf --unset section key`** to unset the value for the key in the specified section:

```
[user@host ~]$ sudo nfsconf --unset nfsd vers4.2
```

Configure an NFSv4-only Client

You can configure an NFSv4-only client by setting the following values on the **`/etc/nfs.conf`** configuration file.

Start by disabling UDP and other NFSv2 and NFSv3 related keys:

```
[user@host ~]$ sudo nfsconf --set nfsd udp n
[user@host ~]$ sudo nfsconf --set nfsd vers2 n
[user@host ~]$ sudo nfsconf --set nfsd vers3 n
```

Enable TCP, and NFSv4, related keys.

```
[user@host ~]$ sudo nfsconf --set nfsd tcp y
[user@host ~]$ sudo nfsconf --set nfsd vers4 y
[user@host ~]$ sudo nfsconf --set nfsd vers4.0 y
[user@host ~]$ sudo nfsconf --set nfsd vers4.1 y
[user@host ~]$ sudo nfsconf --set nfsd vers4.2 y
```

As before, the changes appear in the **`/etc/nfs.conf`** configuration file:

```
[[user@host ~]$ cat /etc/nfs.conf
[nfsd]
udp = n
vers2 = n
vers3 = n
tcp = y
vers4 = y
vers4.0 = y
vers4.1 = y
vers4.2 = y
```

REFERENCES

`mount`(8), **`umount`**(8), **`fstab`**(5), **`mount.nfs`**(8), **`nfs.conf`**(8) and **`nfsconf`**(8) man pages

GUIDED EXERCISE

MANAGING NETWORK-ATTACHED STORAGE WITH NFS

PERFORMANCE CHECKLIST

In this exercise, you will modify the **/etc/fstab** file to persistently mount an NFS export at boot time.

OUTCOMES

You should be able to:

- Test an NFS Server using the **mount** command.
- Configure NFS Shares in the **/etc/fstab** configuration file to save changes even after a system reboots.
- Configure NFS clients to use NFSv4 using the new **nfsconf** tool.

BEFORE YOU BEGIN

Log in to workstation as student using student as the password.

On workstation, run the **lab netstorage-nfs start** command. This command runs a start script that determines if the servera and serverb machines are reachable on the network. The script will alert you if they are not available. The start script configures serverb as an NFSv4 Server, sets up permissions, and exports directories. It creates users and groups needed on both servera and serverb.

```
[student@workstation ~]$ lab netstorage-nfs start
```

A shipping company uses a central server, serverb, to host a number of shared documents and directories. Users on servera, who are all members of the admin group, need access to the persistently mounted NFS share.

Important information:

- serverb shares the **/shares/public** directory, which contains some text files.
- Members of the admin group (admin1, sysmanager1) have read and write access to the **/shares/public** shared directory.
- The principle mount point for servera is **/public**.
- All user passwords are set to redhat.

1. Log in to servera as the student user and switch to the root user.

   1.1. Log in to servera as the student user.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

1.2. Use the **sudo -i** command to switch to the `root` user. The password for the `student` user is `student`.

```
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

2. Use the **nfsconf** tool to configure **/etc/nfs.conf** to enable NFS Clients to work in version 4.X only and to ensure that TCP mode is enabled and UDP mode is disabled.

2.1. Use the **nfsconf** tool to disable the keys `udp`, `vers2`, `vers3`.

```
[root@servera ~]# nfsconf --set nfsd udp n
[root@servera ~]# nfsconf --set nfsd vers2 n
[root@servera ~]# nfsconf --set nfsd vers3 n
```

2.2. Use the **nfsconf** tool to enable the keys `tcp`, `vers4`, `vers4.0`, `vers4.1`, `vers4.2`.

```
[root@servera ~]# nfsconf --set nfsd tcp y
[root@servera ~]# nfsconf --set nfsd vers4 y
[root@servera ~]# nfsconf --set nfsd vers4.0 y
[root@servera ~]# nfsconf --set nfsd vers4.1 y
[root@servera ~]# nfsconf --set nfsd vers4.2 y
```

3. Test the NFS server on `serverb` using `servera` as the NFS client.

3.1. Create the mount point **/public** on `servera`.

```
[root@servera ~]# mkdir /public
```

3.2. On `servera`, use the **mount** command to verify that the **/share/public** NFS share exported by `serverb` mounts correctly on the **/public** mount point.

```
[root@servera ~]# mount -t nfs serverb.lab.example.com:/shares/
public /public
```

3.3. List the contents of the mounted NFS share.

```
[root@servera ~]# ls -l /public
total 16
-rw-r--r--. 1 root admin 42 Apr  8 22:36 Delivered.txt
-rw-r--r--. 1 root admin 46 Apr  8 22:36 NOTES.txt
-rw-r--r--. 1 root admin 20 Apr  8 22:36 README.txt
-rw-r--r--. 1 root admin 27 Apr  8 22:36 Trackings.txt
```

3.4. Explore the **mount** options for the NFS mounted share.

```
[root@servera ~]# mount | grep public
serverb.lab.example.com:/shares/public on /public type nfs4
(rw,relatime,vers=4.2,rsize=262144,wsize=262144,namlen=255,sync,proto=tcp,timeo=600,
retrans=2,sec=sys,clientaddr=172.25.250.10,local_lock=none,addr=172.25.250.11)
```

3.5. Unmount the NFS share.

```
[root@servera ~]# umount /public
```

4. Configure `servera` to ensure that the share used above is persistently mounted.

4.1. Open the **`/etc/fstab`** file for editing.

```
[root@servera ~]# vim /etc/fstab
```

Add the following line to the end of the file:

```
serverb.lab.example.com:/shares/public  /public  nfs  rw,sync  0 0
```

4.2. Use the **`mount`** command to mount the shared directory.

```
[root@servera ~]# mount /public
```

4.3. List the contents of the shared directory.

```
[root@servera ~]# ls -l /public
total 16
-rw-r--r--. 1 root   admin 42 Apr  8 22:36 Delivered.txt
-rw-r--r--. 1 root   admin 46 Apr  8 22:36 NOTES.txt
-rw-r--r--. 1 root   admin 20 Apr  8 22:36 README.txt
-rw-r--r--. 1 root   admin 27 Apr  8 22:36 Trackings.txt
```

4.4. Reboot the `servera` machine.

```
[root@servera ~]# systemctl reboot
```

5. After `servera` has finished rebooting, log in to `servera` as the `admin1` user and test the persistently mounted NFS share.

5.1. Log in to `servera` as the `admin1` user.

```
[student@workstation ~]$ ssh admin1@servera
[admin1@servera ~]$
```

5.2. Test the NFS share mounted on **`/public`**

```
[admin1@servera ~]$ ls -l /public
total 16
-rw-r--r--. 1 root   admin 42 Apr  8 22:36 Delivered.txt
-rw-r--r--. 1 root   admin 46 Apr  8 22:36 NOTES.txt
-rw-r--r--. 1 root   admin 20 Apr  8 22:36 README.txt
-rw-r--r--. 1 root   admin 27 Apr  8 22:36 Trackings.txt
[admin1@servera ~]$ cat /public/NOTES.txt
```

```
###In this file you can log all your notes###
[admin1@servera ~]$ echo "This is a test" > /public/Test.txt
[admin1@servera ~]$ cat /public/Test.txt
This is a test
```

5.3. Log off from `servera`.

```
[admin1@servera ~]$ exit
logout
Connection to servera closed.
```

Finish

On `workstation`, run the **`lab netstorage-nfs finish`** script to complete this exercise.

```
[student@workstation ~]$ lab netstorage-nfs finish
```

This concludes the guided exercise.

AUTOMOUNTING NETWORK-ATTACHED STORAGE

OBJECTIVES

After completing this section, you should be able to:

- Describe the benefits of using the automounter.
- Automount NFS shares using direct and indirect maps, including wildcards.

MOUNTING NFS SHARES WITH THE AUTOMOUNTER

The *automounter* is a service (`autofs`) that automatically mounts NFS shares "on-demand," and will automatically unmount NFS shares when they are no longer being used.

Automounter Benefits

- Users do not need to have root privileges to run the **`mount`** and **`umount`** commands.
- NFS shares configured in the automounter are available to all users on the machine, subject to access permissions.
- NFS shares are not permanently connected like entries in **`/etc/fstab`**, freeing network and system resources.
- The automounter is configured on the client side; no server-side configuration is required.
- The automounter uses the same options as the **`mount`** command, including security options.
- The automounter supports both direct and indirect mount-point mapping, for flexibility in mount-point locations.
- `autofs` creates and removes indirect mount points, eliminating manual management.
- NFS is the default automounter network file system, but other network file systems can be automatically mounted.
- `autofs` is a service that is managed like other system services.

Create an automount

Configuring an automount is a multiple step process:

1. Install the *autofs* package.

```
[user@host ~]$ sudo yum install autofs
```

   This package contains everything needed to use the automounter for NFS shares.

2. Add a *master map* file to **`/etc/auto.master.d`**. This file identifies the base directory used for mount points and identifies the mapping file used for creating the automounts.

```
[user@host ~]$ sudo vim /etc/auto.master.d/demo.autofs
```

The name of the master map file is arbitrary (although typically meaningful), but it must have an extension of **`.autofs`** for the subsystem to recognize it. You can place multiple entries in a single master map file; alternatively, you can create multiple master map files each with its own entries grouped logically.

Add the master map entry, in this case, for indirectly mapped mounts:

```
/shares  /etc/auto.demo
```

This entry uses the **`/shares`** directory as the base for indirect automounts. The **`/etc/auto.demo`** file contains the mount details. Use an absolute file name. The **`auto.demo`** file needs to be created before starting the `autofs` service.

3. Create the mapping files. Each mapping file identifies the mount point, mount options, and source location to mount for a set of automounts.

```
[user@host ~]$ sudo vim /etc/auto.demo
```

The mapping file-naming convention is **`/etc/auto.`*`name`***, where *name* reflects the content of the map.

```
work  -rw,sync  serverb:/shares/work
```

The format of an entry is *mount point*, *mount options*, and *source location*. This example shows a basic indirect mapping entry. Direct maps and indirect maps using wildcards are covered later in this section.

- Known as the *key* in the man pages, the *mount point* is created and removed automatically by the `autofs` service. In this case, the fully qualified mount point is **`/shares/work`** (see the master map file). The **`/shares`** directory and the **`/shares/work`** directories are created and removed as needed by the `autofs` service.

  In this example, the local mount point mirrors the server's directory structure, however this is not required; the local mount point can be named anything. The `autofs` service does not enforce a specific naming structure on the client.

- Mount options start with a dash character (-) and are comma-separated with no white space. Mount options available to a manual mounting of a file system are available when automounting. In this example, the automounter mounts the share with read/write access (**`rw`** option), and the server is synchronized immediately during write operations (**`sync`** option).

  Useful automounter-specific options include **`-fstype=`** and **`-strict`**. Use **`fstype`** to specify the file system type, for example, **`nfs4`** or **`xfs`**, and use **`strict`** to treat errors when mounting file systems as fatal.

- The source location for NFS shares follows the `host:/pathname` pattern; in this example, `serverb:/shares/work`. For this automount to succeed, the NFS server, `serverb`, must *export* the directory with read/write access and the user requesting access must have standard Linux file permissions on the directory. If `serverb` exports the directory with read/only access, then the client will get read/only access even though it requested read/write access.

4. Start and enable the automounter service.

Use **systemctl** to start and enable the autofs service.

```
[user@host ~]$ sudo systemctl enable --now autofs
Created symlink /etc/systemd/system/multi-user.target.wants/autofs.service → /
usr/lib/systemd/system/autofs.service.
```

Direct Maps

Direct maps are used to map an NFS share to an existing absolute path mount point.

To use directly mapped mount points, the master map file might appear as follows:

```
/-  /etc/auto.direct
```

All direct map entries use **/-** as the base directory. In this case, the mapping file that contains the mount details is **/etc/auto.direct**.

The content for the **/etc/auto.direct** file might appear as follows:

```
/mnt/docs  -rw,sync  serverb:/shares/docs
```

The mount point (or key) is always an absolute path. The rest of the mapping file uses the same structure.

In this example the **/mnt** directory exists, and it is not managed by autofs. The full directory **/mnt/docs** will be created and removed automatically by the autofs service.

Indirect Wildcard Maps

When an NFS server exports multiple subdirectories within a directory, then the automounter can be configured to access any one of those subdirectories using a single mapping entry.

Continuing the previous example, if serverb:/shares exports two or more subdirectories and they are accessible using the same mount options, then the content for the **/etc/auto.demo** file might appear as follows:

```
*  -rw,sync  serverb:/shares/&
```

The mount point (or key) is an asterisk character (*), and the subdirectory on the source location is an ampersand character (&). Everything else in the entry is the same.

When a user attempts to access **/shares/work**, the key ***** (which is **work** in this example) replaces the ampersand in the source location and serverb:/shares/work is mounted. As with the indirect example, the **work** directory is created and removed automatically by autofs.

REFERENCES

autofs(5), **automount**(8), **auto.master**(5), and **mount.nfs**(8) man pages

GUIDED EXERCISE

AUTOMOUNTING NETWORK-ATTACHED STORAGE

PERFORMANCE CHECKLIST

In this exercise, you will create direct mapped and indirect mapped automount-managed mount points that mount NFS file systems.

OUTCOMES

You should be able to:

- Install required packages needed for the automounter.
- Configure direct and indirect automounter maps, getting resources from a preconfigured NFSv4 server.
- Understand the difference between direct and indirect automounter maps.

BEFORE YOU BEGIN

Log in to `workstation` as `student` using `student` as the password.

On `workstation`, run the **`lab netstorage-autofs start`** command. This start script determines if `servera` and `serverb` are reachable on the network. The script will alert you if they are not available. The start script configures `serverb` as an NFSv4 server, sets up permissions, and exports directories. It also creates users and groups needed on both `servera` and `serverb`.

```
[student@workstation ~]$ lab netstorage-autofs start
```

An internet service provider uses a central server, `serverb`, to host shared directories containing important documents that need to be available on demand. When users log in to `servera` they need access to the automounted shared directories.

Important information:

- `serverb` is exporting as an NFS share the **`/shares/indirect`** directory, which in turn contains the **`west`**, **`central`**, and **`east`** subdirectories.
- `serverb` is also exporting as an NFS share the **`/shares/direct/external`** directory.
- The `operators` group consists of the `operator1` and `operator2` users. They have read and write access to the shared directories **`/shares/indirect/west`**, **`/shares/indirect/central`**, and **`/shares/indirect/east`**.
- The `contractors` group consists of the `contractor1` and `contractor2` users. They have read and write access to the **`/shares/direct/external`** shared directory.
- The expected mount points for `servera` are **`/external`** and **`/internal`**.

- The **/shares/direct/external** shared directory should be automounted on servera using a *direct* map on **/external**.
- The **/shares/indirect/west** shared directory should be automounted on servera using an *indirect* map on **/internal/west**.
- The **/shares/indirect/central** shared directory should be automounted on servera using an *indirect* map on **/internal/central**.
- The **/shares/indirect/east** shared directory should be automounted on servera using an *indirect* map on **/internal/east**.
- All user passwords are set to redhat.

1. Log in to servera and install the required packages.

   1.1. Log in to servera as the student user.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

   1.2. Use the **sudo -i** command to switch to the root user. The password for the student user is student.

```
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

   1.3. Install the *autofs* package.

```
[root@servera ~]# yum install autofs
...output omitted...
Is this ok [y/N]: y
...output omitted...
```

2. Configure an automounter direct map on servera using shares from serverb. Create the direct map using files named **/etc/auto.master.d/direct.autofs** for the master map and **/etc/auto.direct** for the mapping file. Use the **/external** directory as the main mount point on servera.

   2.1. Test the NFS server and share before proceeding to configure the automounter.

```
[root@servera ~]# mount -t nfs \
serverb.lab.example.com:/shares/direct/external /mnt
[root@servera ~]# ls -l /mnt
total 4
-rw-r--r--. 1 root contractors 22 Apr  7 23:15 README.txt
[root@servera ~]# umount /mnt
```

   2.2. Create a master map file named **/etc/auto.master.d/direct.autofs**, insert the following content, and save the changes.

```
[root@servera ~]# vim /etc/auto.master.d/direct.autofs
/- /etc/auto.direct
```

2.3. Create a direct map file named **/etc/auto.direct**, insert the following content, and save the changes.

```
[root@servera ~]# vim /etc/auto.direct
/external -rw,sync,fstype=nfs4 serverb.lab.example.com:/shares/direct/
external
```

3. Configure an automounter indirect map on `servera` using shares from `serverb`. Create the indirect map using files named **/etc/auto.master.d/indirect.autofs** for the master map and **/etc/auto.indirect** for the mapping file. Use the **/internal** directory as the main mount point on `servera`.

3.1. Test the NFS server and share before proceeding to configure the automounter.

```
[root@servera ~]# mount -t nfs serverb.lab.example.com:/shares/
indirect /mnt
[root@servera ~]# ls -l /mnt
total 0
drwxrws---. 2 root operators 24 Apr  7 23:34 central
drwxrws---. 2 root operators 24 Apr  7 23:34 east
drwxrws---. 2 root operators 24 Apr  7 23:34 west
[root@servera ~]# umount /mnt
```

3.2. Create a master map file named **/etc/auto.master.d/indirect.autofs**, insert the following content, and save the changes.

```
[root@servera ~]# vim /etc/auto.master.d/indirect.autofs
/internal /etc/auto.indirect
```

3.3. Create an indirect map file named **/etc/auto.indirect**, insert the following content, and save the changes.

```
[root@servera ~]# vim /etc/auto.indirect
* -rw,sync,fstype=nfs4 serverb.lab.example.com:/shares/indirect/&
```

4. Start the `autofs` service on `servera` and enable it to start automatically at boot time. Reboot `servera` to determine if the `autofs` service starts automatically.

4.1. Start and enable the `autofs` service on `servera`.

```
[root@servera ~]# systemctl enable --now autofs
Created symlink /etc/systemd/system/multi-user.target.wants/
autofs.service → /usr/lib/systemd/system/autofs.service.
```

4.2. Reboot the `servera` machine.

```
[root@servera ~]# systemctl reboot
```

5. Test the direct automounter map as the `contractor1` user. When done, exit from the `contractor1` user session on `servera`.

5.1. After the `servera` machine has finished booting, log in to `servera` as the `student` user.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

5.2. Switch to the `contractor1` user.

```
[student@servera ~]$ su - contractor1
Password: redhat
```

5.3. List the **`/external`** mount point.

```
[contractor1@servera ~]$ ls -l /external
total 4
-rw-r--r--. 1 root contractors 22 Apr  7 23:34 README.txt
```

5.4. Review the content and test the access to the **`/external`** mount point.

```
[contractor1@servera ~]$ cat /external/README.txt
###External Folder###
[contractor1@servera ~]$ echo testing-direct > /external/testing.txt
[contractor1@servera ~]$ cat /external/testing.txt
testing-direct
```

5.5. Exit from the `contractor1` user session.

```
[contractor1@servera ~]$ exit
logout
```

6. Test the indirect automounter map as the `operator1` user. When done, log off from `servera`.

6.1. Switch to `operator1` user.

```
[student@servera ~]$ su - operator1
Password: redhat
```

6.2. List the **`/internal`** mount point.

```
[operator1@servera ~]$ ls -l /internal
total 0
```

NOTE

You will notice that in an automounter indirect map, even if you are in the mapped mount point, you need to call each of the shared subdirectories or files on demand to get access to them. In an automounter direct map, after you open the mapped mount point, you get access to the directories and content configured in the shared directory.

6.3. Test the **`/internal/west`** automounter shared directory access.

```
[operator1@servera ~]$ ls -l /internal/west/
total 4
-rw-r--r--. 1 root operators 18 Apr  7 23:34 README.txt
[operator1@servera ~]$ cat /internal/west/README.txt
###West Folder###
[operator1@servera ~]$ echo testing-1 > /internal/west/testing-1.txt
[operator1@servera ~]$ cat /internal/west/testing-1.txt
testing-1
[operator1@servera ~]$ ls -l /internal
total 0
drwxrws---. 2 root operators 24 Apr  7 23:34 west
```

6.4. Test the **`/internal/central`** automounter shared directory access.

```
[operator1@servera ~]$ ls -l /internal/central
total 4
-rw-r--r--. 1 root operators 21 Apr  7 23:34 README.txt
[operator1@servera ~]$ cat /internal/central/README.txt
###Central Folder###
[operator1@servera ~]$ echo testing-2 > /internal/central/
testing-2.txt
[operator1@servera ~]$ cat /internal/central/testing-2.txt
testing-2
```

```
[operator1@servera ~]$ ls -l /internal
total 0
drwxrws---. 2 root operators 24 Apr  7 23:34 central
drwxrws---. 2 root operators 24 Apr  7 23:34 west
```

6.5. Test the **/internal/east** automounter shared directory access.

```
[operator1@servera ~]$ ls -l /internal/east
total 4
-rw-r--r--. 1 root operators 18 Apr  7 23:34 README.txt
[operator1@servera ~]$ cat /internal/east/README.txt
###East Folder###
[operator1@servera ~]$ echo testing-3 > /internal/east/testing-3.txt
[operator1@servera ~]$ cat /internal/east/testing-3.txt
testing-3
[operator1@servera ~]$ ls -l /internal
total 0
drwxrws---. 2 root operators 24 Apr  7 23:34 central
drwxrws---. 2 root operators 24 Apr  7 23:34 east
drwxrws---. 2 root operators 24 Apr  7 23:34 west
```

6.6. Test the **/external** automounter shared directory access.

```
[operator1@servera ~]$ ls -l /external
ls: cannot open directory '/external': Permission denied
```

6.7. Log off from `servera`.

```
[operator1@servera ~]$ exit
logout
[student@servera ~]$ exit
logout
Connection to servera closed.
```

Finish

On `workstation`, run the **lab netstorage-autofs finish** script to complete this exercise.

```
[student@workstation ~]$ lab netstorage-autofs finish
```

This concludes the guided exercise.

LAB

ACCESSING NETWORK-ATTACHED STORAGE

PERFORMANCE CHECKLIST

In this lab, you will set up the automounter with an indirect map, using shares from an NFSv4 server.

OUTCOMES

You should be able to:

- Install required packages needed to set up the automounter.
- Configure an automounter indirect map, getting resources from a preconfigured NFSv4 server.
- Configure an NFS client to use NFSv4 using the **`nfsconf`** tool.

BEFORE YOU BEGIN

Log in to `workstation` as `student` using `student` as the password.

On `workstation`, run the **`lab netstorage-review start`** command. This start script determines if the `servera` and `serverb` systems are reachable on the network. The start script configures `serverb` as an NFSv4 server, sets up permissions, and exports directories. It also creates users and groups needed on both `servera` and `serverb` systems.

```
[student@workstation ~]$ lab netstorage-review start
```

An IT support company uses a central server, `serverb`, to host some shared directories on **`/remote/shares`** for their groups and users. Users need to be able to log in and have their shared directories mounted on demand and ready to use, under the **`/shares`** directory on `servera`.

Important information:

- `serverb` is sharing the **`/shares`** directory, which in turn contains the **`management`**, **`production`** and **`operation`** subdirectories.
- The `managers` group consists of the `manager1` and `manager2` users. They have read and write access to the **`/shares/management`** shared directory.
- The `production` group consists of the `dbuser1` and `sysadmin1` users. They have read and write access to the **`/shares/production`** shared directory.
- The `operators` group consists of the `contractor1` and `consultant1` users. They have read and write access to the **`/shares/operation`** shared directory.
- The main mount point for `servera` is the **`/remote`** directory.

- The **/shares/management** shared directory should be automounted on **/remote/management** on servera.
- The **/shares/production** shared directory should be automounted on **/remote/production** on servera.
- The **/shares/operation** shared directory should be automounted on **/remote/operation** on servera.
- All user passwords are set to redhat.

1. Log in to servera and install the required packages.
2. Use the **nfsconf** command to configure **/etc/nfs.conf**. Enable the NFS client to work only in version 4.X and ensure that TCP mode is enabled and UDP mode is disabled.
3. Configure an automounter indirect map on servera using shares from serverb. Create an indirect map using files named **/etc/auto.master.d/shares.autofs** for the master map and **/etc/auto.shares** for the mapping file. Use the **/remote** directory as the main mount point on servera. Reboot servera to determine if the autofs service starts automatically.
4. Test the autofs configuration with the various users. When done, log off from servera.

Evaluation

On workstation, run the **lab netstorage-review grade** command to confirm success of this exercise.

```
[student@workstation ~]$ lab netstorage-review grade
```

Finish

On workstation, run the **lab netstorage-review finish** command to complete this exercise.

```
[student@workstation ~]$ lab netstorage-review finish
```

This concludes the lab.

SOLUTION

ACCESSING NETWORK-ATTACHED STORAGE

PERFORMANCE CHECKLIST

In this lab, you will set up the automounter with an indirect map, using shares from an NFSv4 server.

OUTCOMES

You should be able to:

- Install required packages needed to set up the automounter.
- Configure an automounter indirect map, getting resources from a preconfigured NFSv4 server.
- Configure an NFS client to use NFSv4 using the **`nfsconf`** tool.

BEFORE YOU BEGIN

Log in to `workstation` as `student` using `student` as the password.

On `workstation`, run the **`lab netstorage-review start`** command. This start script determines if the `servera` and `serverb` systems are reachable on the network. The start script configures `serverb` as an NFSv4 server, sets up permissions, and exports directories. It also creates users and groups needed on both `servera` and `serverb` systems.

```
[student@workstation ~]$ lab netstorage-review start
```

An IT support company uses a central server, `serverb`, to host some shared directories on **`/remote/shares`** for their groups and users. Users need to be able to log in and have their shared directories mounted on demand and ready to use, under the **`/shares`** directory on `servera`.

Important information:

- `serverb` is sharing the **`/shares`** directory, which in turn contains the **`management`**, **`production`** and **`operation`** subdirectories.
- The `managers` group consists of the `manager1` and `manager2` users. They have read and write access to the **`/shares/management`** shared directory.
- The `production` group consists of the `dbuser1` and `sysadmin1` users. They have read and write access to the **`/shares/production`** shared directory.
- The `operators` group consists of the `contractor1` and `consultant1` users. They have read and write access to the **`/shares/operation`** shared directory.
- The main mount point for `servera` is the **`/remote`** directory.

- The **/shares/management** shared directory should be automounted on **/remote/management** on servera.
- The **/shares/production** shared directory should be automounted on **/remote/production** on servera.
- The **/shares/operation** shared directory should be automounted on **/remote/operation** on servera.
- All user passwords are set to redhat.

1. Log in to servera and install the required packages.

   1.1. Log in to servera as the student user.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

   1.2. Use the **sudo -i** command to switch to the root user. The password for the student user is student.

```
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

   1.3. Install the *autofs* package.

```
[root@servera ~]# yum install autofs
...output omitted...
Is this ok [y/N]: y
...output omitted...
```

2. Use the **nfsconf** command to configure **/etc/nfs.conf**. Enable the NFS client to work only in version 4.X and ensure that TCP mode is enabled and UDP mode is disabled.

   2.1. Use the **nfsconf** tool to disable the keys udp, vers2, vers3.

```
[root@servera ~]# nfsconf --set nfsd udp n
[root@servera ~]# nfsconf --set nfsd vers2 n
[root@servera ~]# nfsconf --set nfsd vers3 n
```

   2.2. Use the **nfsconf** tool to enable the keys tcp, vers4, vers4.0, vers4.1, vers4.2.

```
[root@servera ~]# nfsconf --set nfsd tcp y
[root@servera ~]# nfsconf --set nfsd vers4 y
[root@servera ~]# nfsconf --set nfsd vers4.0 y
[root@servera ~]# nfsconf --set nfsd vers4.1 y
[root@servera ~]# nfsconf --set nfsd vers4.2 y
```

3. Configure an automounter indirect map on servera using shares from serverb. Create an indirect map using files named **/etc/auto.master.d/shares.autofs** for the master map and **/etc/auto.shares** for the mapping file. Use the **/remote** directory as the

main mount point on `servera`. Reboot `servera` to determine if the `autofs` service starts automatically.

3.1. Test the NFS server before proceeding to configure the automounter.

```
[root@servera ~]# mount -t nfs serverb.lab.example.com:/shares /mnt
[root@servera ~]# ls -l /mnt
total 0
drwxrwx---. 2 root managers   25 Apr  4 01:13 management
drwxrwx---. 2 root operators  25 Apr  4 01:13 operation
drwxrwx---. 2 root production 25 Apr  4 01:13 production
[root@servera ~]# umount /mnt
```

3.2. Create a master map file named **`/etc/auto.master.d/shares.autofs`**, insert the following content, and save the changes.

```
[root@servera ~]# vim /etc/auto.master.d/shares.autofs
/remote /etc/auto.shares
```

3.3. Create an indirect map file named **`/etc/auto.shares`**, insert the following content, and save the changes.

```
[root@servera ~]# vim /etc/auto.shares
* -rw,sync,fstype=nfs4 serverb.lab.example.com:/shares/&
```

3.4. Start and enable the `autofs` service on `servera`.

```
[root@servera ~]# systemctl enable --now autofs
Created symlink /etc/systemd/system/multi-user.target.wants/
autofs.service → /usr/lib/systemd/system/autofs.service.
```

3.5. Reboot the`servera` machine.

```
[root@servera ~]# systemctl reboot
```

4. Test the `autofs` configuration with the various users. When done, log off from `servera`.

4.1. After the `servera` machine has finished booting, log in to `servera` as the `student` user.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

4.2. Use the **`su - manager1`** command to switch to the `manager1` user and test access.

```
[student@servera ~]$ su - manager1
Password: redhat
[manager1@servera ~]$ ls -l /remote/management/
total 4
-rw-r--r--. 1 root managers 46 Apr  4 01:13 Welcome.txt
```

```
[manager1@servera ~]$ cat /remote/management/Welcome.txt
###Welcome to Management Folder on SERVERB###
[manager1@servera ~]$ echo TEST1 > /remote/management/Test.txt
[manager1@servera ~]$ cat /remote/management/Test.txt
TEST1
[manager1@servera ~]$ ls -l /remote/operation/
ls: cannot open directory '/remote/operation/': Permission denied
[manager1@servera ~]$ ls -l /remote/production/
ls: cannot open directory '/remote/production/': Permission denied
[manager1@servera ~]$ exit
logout
```

4.3. Switch to the `dbuser1` user and test access.

```
[student@servera ~]$ su - dbuser1
Password: redhat
[dbuser1@servera ~]$ ls -l /remote/production/
total 4
-rw-r--r--. 1 root production 46 Apr  4 01:13 Welcome.txt
[dbuser1@servera ~]$ cat /remote/production/Welcome.txt
###Welcome to Production Folder on SERVERB###
[dbuser1@servera ~]$ echo TEST2 > /remote/production/Test.txt
[dbuser1@servera ~]$ cat /remote/production/Test.txt
TEST2
[dbuser1@servera ~]$ ls -l /remote/operation/
ls: cannot open directory '/remote/operation/': Permission denied
[dbuser1@servera ~]$ ls -l /remote/management/
ls: cannot open directory '/remote/management/': Permission denied
[dbuser1@servera ~]$ exit
logout
```

4.4. Switch to the `contractor1` user and test access.

```
[student@servera ~]$ su - contractor1
Password: redhat
[contractor1@servera ~]$ ls -l /remote/operation/
total 4
-rw-r--r--. 1 root operators 45 Apr  4 01:13 Welcome.txt
[contractor1@servera ~]$ cat /remote/operation/Welcome.txt
###Welcome to Operation Folder on SERVERB###
[contractor1@servera ~]$ echo TEST3 > /remote/operation/Test.txt
[contractor1@servera ~]$ cat /remote/operation/Test.txt
TEST3
[contractor1@servera ~]$ ls -l /remote/management/
ls: cannot open directory '/remote/management/': Permission denied
```

```
[contractor1@servera ~]$ ls -l /remote/production/
ls: cannot open directory '/remote/production/': Permission denied
[contractor1@servera ~]$ exit
logout
```

4.5. Explore the **mount** options for the NFS automounted share.

```
[student@servera ~]$ mount | grep nfs
rpc_pipefs on /var/lib/nfs/rpc_pipefs type rpc_pipefs (rw,relatime)
serverb.lab.example.com:/shares/management on /remote/management
 type nfs4
(rw,relatime,vers=4.2,rsize=262144,wsize=262144,namlen=255,sync,proto=tcp,timeo=600,
retrans=2,sec=sys,clientaddr=172.25.250.10,local_lock=none,addr=172.25.250.11)
serverb.lab.example.com:/shares/operation on /remote/operation
 type nfs4
(rw,relatime,vers=4.2,rsize=262144,wsize=262144,namlen=255,sync,proto=tcp,timeo=600,
retrans=2,sec=sys,clientaddr=172.25.250.10,local_lock=none,addr=172.25.250.11)
serverb.lab.example.com:/shares/production on /remote/production
 type nfs4
(rw,relatime,vers=4.2,rsize=262144,wsize=262144,namlen=255,sync,proto=tcp,timeo=600,
retrans=2,sec=sys,clientaddr=172.25.250.10,local_lock=none,addr=172.25.250.11)
```

4.6. Log off from `servera`.

```
[student@servera ~]$ exit
logout
```

Evaluation

On `workstation`, run the **`lab netstorage-review grade`** command to confirm success of this exercise.

```
[student@workstation ~]$ lab netstorage-review grade
```

Finish

On `workstation`, run the **`lab netstorage-review finish`** command to complete this exercise.

```
[student@workstation ~]$ lab netstorage-review finish
```

This concludes the lab.

SUMMARY

In this chapter, you learned how to:

- Mount and unmount an NFS export from the command line.
- Configure an NFS export to automatically mount at startup.
- Configure the automounter with direct and indirect maps, and describe their differences.
- Configure NFS clients to use NFSv4 using the new **`nfsconf`** tool.

CHAPTER 10

CONTROLLING THE BOOT PROCESS

GOAL

Manage the boot process to control services offered and to troubleshoot and repair problems.

OBJECTIVES

- Describe the Red Hat Enterprise Linux boot process, set the default target used when booting, and boot a system to a non-default target.
- Log in to a system and change the root password when the current root password has been lost.
- Manually repair file system configuration or corruption issues that stop the boot process.

SECTIONS

- Selecting the Boot Target (and Guided Exercise)
- Resetting the Root Password (and Guided Exercise)
- Repairing File System Issues at Boot (and Guided Exercise)

LAB

Controlling the Boot Process

SELECTING THE BOOT TARGET

OBJECTIVES

After completing this section, you should be able to:

- Describe the Red Hat Enterprise Linux boot process.
- Set the default target used when booting.
- Boot a system to a non-default target.

DESCRIBING THE RED HAT ENTERPRISE LINUX 8 BOOT PROCESS

Modern computer systems are complex combinations of hardware and software. Starting from an undefined, powered-down state to a running system with a login prompt requires a large number of pieces of hardware and software to work together. The following list gives a high-level overview of the tasks involved for a physical **x86_64** system booting Red Hat Enterprise Linux 8. The list for **x86_64** virtual machines is roughly the same, but the hypervisor handles some of the hardware-specific steps in software.

- The machine is powered on. The system firmware, either modern UEFI or older BIOS, runs a *Power On Self Test (POST)* and starts to initialize some of the hardware.

  Configured using the system BIOS or UEFI configuration screens that you typically reach by pressing a specific key combination, such as **F2**, early during the boot process.

- The system firmware searches for a bootable device, either configured in the UEFI boot firmware or by searching for a *Master Boot Record (MBR)* on all disks, in the order configured in the BIOS.

  Configured using the system BIOS or UEFI configuration screens that you typically reach by pressing a specific key combination, such as **F2**, early during the boot process.

- The system firmware reads a boot loader from disk and then passes control of the system to the boot loader. On a Red Hat Enterprise Linux 8 system, the boot loader is the *GRand Unified Bootloader version 2 (GRUB2)*.

  Configured using the **`grub2-install`** command, which installs GRUB2 as the boot loader on the disk.

- GRUB2 loads its configuration from the **`/boot/grub2/grub.cfg`** file and displays a menu where you can select which kernel to boot.

  Configured using the **`/etc/grub.d/`** directory, the **`/etc/default/grub`** file, and the **`grub2-mkconfig`** command to generate the **`/boot/grub2/grub.cfg`** file.

- After you select a kernel, or the timeout expires, the boot loader loads the kernel and *initramfs* from disk and places them in memory. An **`initramfs`** is an archive containing the kernel modules for all the hardware required at boot, initialization scripts, and more. On Red Hat Enterprise Linux 8, the **`initramfs`** contains an entire usable system by itself.

Configured using the **/etc/dracut.conf.d/** directory, the **dracut** command, and the **lsinitrd** command to inspect the **initramfs** file.

- The boot loader hands control over to the kernel, passing in any options specified on the kernel command line in the boot loader, and the location of the **initramfs** in memory.

  Configured using the **/etc/grub.d/** directory, the **/etc/default/grub** file, and the **grub2-mkconfig** command to generate the **/boot/grub2/grub.cfg** file.

- The kernel initializes all hardware for which it can find a driver in the **initramfs**, then executes **/sbin/init** from the **initramfs** as PID 1. On Red Hat Enterprise Linux 8, **/sbin/init** is a link to **systemd**.

  Configured using the kernel **init=** command-line parameter.

- The `systemd` instance from the **initramfs** executes all units for the `initrd.target` target. This includes mounting the root file system on disk on to the **/sysroot** directory.

  Configured using **/etc/fstab**

- The kernel switches (pivots) the root file system from **initramfs** to the root file system in **/sysroot**. `systemd` then re-executes itself using the copy of `systemd` installed on the disk.

- `systemd` looks for a default target, either passed in from the kernel command line or configured on the system, then starts (and stops) units to comply with the configuration for that target, solving dependencies between units automatically. In essence, a `systemd` target is a set of units that the system should activate to reach the desired state. These targets typically start a text-based login or a graphical login screen.

  Configured using **/etc/systemd/system/default.target** and **/etc/systemd/system/**.

REBOOTING AND SHUTTING DOWN

To power off or reboot a running system from the command line, you can use the **systemctl** command.

systemctl poweroff stops all running services, unmounts all file systems (or remounts them read-only when they cannot be unmounted), and then powers down the system.

systemctl reboot stops all running services, unmounts all file systems, and then reboots the system.

You can also use the shorter version of these commands, **poweroff** and **reboot**, which are symbolic links to their **systemctl** equivalents.

> **NOTE**
> **systemctl halt** and **halt** are also available to stop the system, but unlike **poweroff**, these commands do not power off the system; they bring a system down to a point where it is safe to power it off manually.

SELECTING A SYSTEMD TARGET

A `systemd` target is a set of `systemd` units that the system should start to reach a desired state. The following table lists the most important targets.

Commonly Used Targets

TARGET	PURPOSE
`graphical.target`	System supports multiple users, graphical- and text-based logins.
`multi-user.target`	System supports multiple users, text-based logins only.
`rescue.target`	**`sulogin`** prompt, basic system initialization completed.
`emergency.target`	**`sulogin`** prompt, **`initramfs`** pivot complete, and system root mounted on **`/`** read only.

A target can be a part of another target. For example, the `graphical.target` includes `multi-user.target`, which in turn depends on `basic.target` and others. You can view these dependencies with the following command.

```
[user@host ~]$ systemctl list-dependencies graphical.target | grep target
graphical.target
* └─multi-user.target
*   ├─basic.target
*   │ ├─paths.target
*   │ ├─slices.target
*   │ ├─sockets.target
*   │ ├─sysinit.target
*   │ │ ├─cryptsetup.target
*   │ │ ├─local-fs.target
*   │ │ └─swap.target
```

To list the available targets, use the following command.

```
[user@host ~]$ systemctl list-units --type=target --all
  UNIT                      LOAD      ACTIVE   SUB    DESCRIPTION
  ----------------------------------------------------------------------------
  basic.target              loaded    active   active Basic System
  cryptsetup.target         loaded    active   active Local Encrypted Volumes
  emergency.target          loaded    inactive dead   Emergency Mode
  getty-pre.target          loaded    inactive dead   Login Prompts (Pre)
  getty.target              loaded    active   active Login Prompts
  graphical.target          loaded    inactive dead   Graphical Interface
```

Selecting a Target at Runtime

On a running system, administrators can switch to a different target using the **`systemctl isolate`** command.

```
[root@host ~]# systemctl isolate multi-user.target
```

Isolating a target stops all services not required by that target (and its dependencies), and starts any required services not yet started.

Not all targets can be isolated. You can only isolate targets that have `AllowIsolate=yes` set in their unit files. For example, you can isolate the graphical target, but not the cryptsetup target.

```
[user@host ~]$ systemctl cat graphical.target
# /usr/lib/systemd/system/graphical.target
...output omitted...
[Unit]
Description=Graphical Interface
Documentation=man:systemd.special(7)
Requires=multi-user.target
Wants=display-manager.service
Conflicts=rescue.service rescue.target
After=multi-user.target rescue.service rescue.target display-manager.service
AllowIsolate=yes
[user@host ~]$ systemctl cat cryptsetup.target
# /usr/lib/systemd/system/cryptsetup.target
...output omitted...
[Unit]
Description=Local Encrypted Volumes
Documentation=man:systemd.special(7)
```

Setting a Default Target

When the system starts, `systemd` activates the `default.target` target. Normally the default target in **`/etc/systemd/system/`** is a symbolic link to either `graphical.target` or `multi-user.target`. Instead of editing this symbolic link by hand, the **`systemctl`** command provides two subcommands to manage this link: **`get-default`** and **`set-default`**.

```
[root@host ~]# systemctl get-default
multi-user.target
[root@host ~]# systemctl set-default graphical.target
Removed /etc/systemd/system/default.target.
Created symlink /etc/systemd/system/default.target -> /usr/lib/systemd/system/
graphical.target.
[root@host ~]# systemctl get-default
graphical.target
```

Selecting a Different Target at Boot Time

To select a different target at boot time, append the **`systemd.unit=`*`target`*`.target`** option to the kernel command line from the boot loader.

For example, to boot the system into a rescue shell where you can change the system configuration with almost no services running, append the following option to the kernel command line from the boot loader.

```
systemd.unit=rescue.target
```

This configuration change only affects a single boot, making it a useful tool for troubleshooting the boot process.

To use this method of selecting a different target, use the following procedure:

1. Boot or reboot the system.
2. Interrupt the boot loader menu countdown by pressing any key (except **`Enter`** which would initiate a normal boot).
3. Move the cursor to the kernel entry that you want to start.
4. Press **`e`** to edit the current entry.
5. Move the cursor to the line that starts with **`linux`**. This is the kernel command line.
6. Append **`systemd.unit=`*`target`*`.target`**. For example, **`systemd.unit=emergency.target`**.
7. Press **`Ctrl`**+**`x`** to boot with these changes.

REFERENCES

`info grub2` (*GNU GRUB manual*)

`bootup`(7), **`dracut.bootup`**(7), **`lsinitrd`**(1), **`systemd.target`**(5), **`systemd.special`**(7), **`sulogin`**(8), and **`systemctl`**(1) man pages

For more information, refer to the *Managing services with systemd* chapter in the *Configuring basic system settings* guide at https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/configuring_basic_system_settings/#managing-services-with-systemd

GUIDED EXERCISE

SELECTING THE BOOT TARGET

In this exercise, you will determine the default target into which a system boots, and boot that system into other targets.

OUTCOMES

You should be able to update the system default target and use a temporary target from the boot loader.

BEFORE YOU BEGIN

Log in as the `student` user on `workstation` using `student` as the password.

On `workstation`, run the **`lab boot-selecting start`** command. This command runs a start script that prepares `workstation` for the exercise.

```
[student@workstation ~]$ lab boot-selecting start
```

1. On `workstation`, open a terminal and confirm that the default target is `graphical.target`.

```
[student@workstation ~]$ systemctl get-default
graphical.target
```

2. On `workstation`, switch to the `multi-user` target manually without rebooting. Use the **`sudo`** command and if prompted, use `student` as the password.

```
[student@workstation ~]$ sudo systemctl isolate multi-user.target
[sudo] password for student: student
```

3. Access a text-based console. Use the **`Ctrl`**+**`Alt`**+**`F1`** key sequence using the relevant button or menu entry. Log in as `root` using `redhat` as the password.

> **NOTE**
> Reminder: If you are using the terminal through a webpage you can click the Show Keyboard icon under your web browser's url bar and then to the right of the machine's IP address.

```
workstation login: root
Password: redhat
[root@workstation ~]#
```

4. Configure `workstation` to automatically boot into the `multi-user` target, and then reboot `workstation` to verify. When done, change the default `systemd` target back to the `graphical` target.

4.1. Use the **`systemctl set-default`** command to set the default target.

```
[root@workstation ~]# systemctl set-default multi-user.target
Removed /etc/systemd/system/default.target.
Created symlink /etc/systemd/system/default.target -> /usr/lib/
systemd/system/multi-user.target.
```

4.2. Reboot `workstation`.

```
[root@workstation ~]# systemctl reboot
```

Notice that after reboot the system presents a text-based console and not a graphical login anymore.

4.3. Log in as `root` using `redhat` as the password.

```
workstation login: root
Password: redhat
Last login: Thu Mar 28 14:50:53 on tty1
[root@workstation ~]#
```

4.4. Set the default `systemd` target back to the `graphical` target.

```
[root@workstation ~]# systemctl set-default graphical.target
Removed /etc/systemd/system/default.target.
Created symlink /etc/systemd/system/default.target -> /usr/lib/
systemd/system/graphical.target.
```

This concludes the first part of the exercise where you practice setting the default `systemd` target.

5. In this second part of the exercise, you are practicing the rescue mode.

   Access the boot loader by rebooting `workstation` again. From within the boot loader menu, boot into the `rescue` target.

   5.1. Initiate the reboot.

   ```
   [root@workstation ~]# systemctl reboot
   ```

   5.2. When the boot loader menu appears, press any key to interrupt the countdown (except **Enter**, which would initiate a normal boot).

   5.3. Use the cursor keys to highlight the default boot loader entry.

   5.4. Press **e** to edit the current entry.

   5.5. Using the cursor keys, navigate to the line that starts with **linux**.

   5.6. Press **End** to move the cursor to the end of the line.

   5.7. Append **systemd.unit=rescue.target** to the end of the line.

   5.8. Press **Ctrl**+**x** to boot using the modified configuration.

   5.9. Log in to rescue mode. The `root` password is `redhat` You may need to hit enter to get a clean prompt.

   ```
   Give root password for maintenance
   (or press Control-D to continue): redhat
   [root@workstation ~]#
   ```

6. Confirm that in rescue mode, the root file system is in read/write mode.

   ```
   [root@workstation ~]# mount
   ...output omitted...
   /dev/vda1 on / type xfs (rw,relatime,seclabel,attr2,inode64,noquota)
   ...output omitted...
   ```

7. Press **Ctrl**+**d** to continue with the boot process.

   The system presents a graphical login. Log in as `student` using `student` as the password.

Finish

On `workstation`, run the **lab boot-selecting finish** script to complete this exercise.

```
[student@workstation ~]$ lab boot-selecting finish
```

This concludes the guided exercise.

RESETTING THE ROOT PASSWORD

OBJECTIVES

After completing this section, you should be able to log in to a system and change the `root` password when the current `root` password has been lost.

RESETTING THE ROOT PASSWORD FROM THE BOOT LOADER

One task that every system administrator should be able to accomplish is resetting a lost `root` password. If the administrator is still logged in, either as an unprivileged user but with full **`sudo`** access, or as `root`, this task is trivial. When the administrator is not logged in, this task becomes slightly more involved.

Several methods exist to set a new `root` password. A system administrator could, for example, boot the system using a Live CD, mount the root file system from there, and edit **`/etc/shadow`**. In this section, we explore a method that does not require the use of external media.

> **NOTE**
> On Red Hat Enterprise Linux 6 and earlier, administrators can boot the system into runlevel 1 to get a `root` prompt. The closest analogs to runlevel 1 on a Red Hat Enterprise Linux 8 machine are the rescue and emergency targets, both of which require the `root` password to log in.

On Red Hat Enterprise Linux 8, it is possible to have the scripts that run from the **`initramfs`** pause at certain points, provide a `root` shell, and then continue when that shell exits. This is mostly meant for debugging, but you can also use this method to reset a lost `root` password.

To access that `root` shell, follow these steps:

1. Reboot the system.
2. Interrupt the boot loader countdown by pressing any key, except **`Enter`**.
3. Move the cursor to the kernel entry to boot.
4. Press **`e`** to edit the selected entry.
5. Move the cursor to the kernel command line (the line that starts with **`linux`**).
6. Append **`rd.break`**. With that option, the system breaks just before the system hands control from the **`initramfs`** to the actual system.
7. Press **`Ctrl`**+**`x`** to boot with the changes.

At this point, the system presents a `root` shell, with the actual root file system on the disk mounted read-only on **`/sysroot`**. Because troubleshooting often requires modification to the root file system, you need to change the root file system to read/write. The following step shows how the **`remount,rw`** option to the **`mount`** command remounts the file system with the new option (**`rw`**) set.

NOTE

Prebuilt images may place multiple **console=** arguments to the kernel to support a wide array of implementation scenarios. Those **console=** arguments indicate the devices to use for console output. The caveat with **rd.break** is that even though the system sends the kernel messages to all the consoles, the prompt ultimately uses whichever console is given last. If you do not get your prompt, you may want to temporarily reorder the **console=** arguments when you edit the kernel command line from the boot loader.

IMPORTANT

The system has not yet enabled SELinux, so any file you create does not have an SELinux context. Some tools, such as the **passwd** command, first create a temporary file, then move it in place of the file they are intended to edit, effectively creating a new file without an SELinux context. For this reason, when you use the **passwd** command with **rd.break**, the **/etc/shadow** file does not get an SELinux context.

To reset the `root` password from this point, use the following procedure:

1. Remount **/sysroot** as read/write.

```
switch_root:/# mount -o remount,rw /sysroot
```

2. Switch into a **chroot** jail, where **/sysroot** is treated as the root of the file-system tree.

```
switch_root:/# chroot /sysroot
```

3. Set a new `root` password.

```
sh-4.4# passwd root
```

4. Make sure that all unlabeled files, including **/etc/shadow** at this point, get relabeled during boot.

```
sh-4.4# touch /.autorelabel
```

5. Type **exit** twice. The first command exits the **chroot** jail, and the second command exits the **initramfs** debug shell.

At this point, the system continues booting, performs a full SELinux relabel, and then reboots again.

INSPECTING LOGS

Looking at the logs of previously failed boots can be useful. If the system journals are persistent across reboots, you can use the **journalctl** tool to inspect those logs.

Remember that by default, the system journals are kept in the **/run/log/journal** directory, which means the journals are cleared when the system reboots. To store journals in the **/**

var/log/journal directory, which persists across reboots, set the `Storage` parameter to **persistent** in **/etc/systemd/journald.conf**.

```
[root@host ~]# vim /etc/systemd/journald.conf
...output omitted...
[Journal]
Storage=persistent
...output omitted...
[root@host ~]# systemctl restart systemd-journald.service
```

To inspect the logs of a previous boot, use the **-b** option of **journalctl**. Without any arguments, the **-b** option only displays messages since the last boot. With a negative number as an argument, it displays the logs of previous boots.

```
[root@host ~]# journalctl -b -1 -p err
```

This command shows all messages rated as an error or worse from the previous boot.

REPAIRING SYSTEMD BOOT ISSUES

To troubleshoot service startup issues at boot time, Red Hat Enterprise Linux 8 makes the following tools available.

Enabling the Early Debug Shell

By enabling the `debug-shell` service with **systemctl enable debug-shell.service**, the system spawns a `root` shell on **TTY9** (**Ctrl**+**Alt**+**F9**) early during the boot sequence. This shell is automatically logged in as `root`, so that administrators can debug the system while the operating system is still booting.

> **WARNING**
> Do not forget to disable the `debug-shell.service` service when you are done debugging, because it leaves an unauthenticated `root` shell open to anyone with local console access.

Using the Emergency and Rescue Targets

By appending either **systemd.unit=rescue.target** or **systemd.unit=emergency.target** to the kernel command line from the boot loader, the system spawns into a rescue or emergency shell instead of starting normally. Both of these shells require the `root` password.

The emergency target keeps the root file system mounted read-only, while the rescue target waits for `sysinit.target` to complete, so that more of the system is initialized, such as the logging service or the file systems. The root user at this point can not make changes to /etc/fstab until the drive is remounted in a read write state **mount -o remount,rw /**

Administrators can use these shells to fix any issues that prevent the system from booting normally; for example, a dependency loop between services, or an incorrect entry in **/etc/fstab**. Exiting from these shells continues with the regular boot process.

Identifying Stuck Jobs

During startup, `systemd` spawns a number of jobs. If some of these jobs cannot complete, they block other jobs from running. To inspect the current job list, administrators can use the **`systemctl list-jobs`** command. Any jobs listed as running must complete before the jobs listed as waiting can continue.

REFERENCES

`dracut.cmdline`(7), **`systemd-journald`**(8), **`journald.conf`**(5), **`journalctl`**(1), and **`systemctl`**(1) man pages

GUIDED EXERCISE

RESETTING THE ROOT PASSWORD

In this exercise, you will reset the `root` password on a system.

OUTCOMES

You should be able to reset a lost `root` password.

BEFORE YOU BEGIN

Log in as the `student` user on `workstation` using `student` as the password.

On `workstation`, run the **`lab boot-resetting start`** command. This command runs a start script that determines if the `servera` machine is reachable on the network. It also resets the `root` password to a random string and sets a higher timeout for the GRUB2 menu.

```
[student@workstation ~]$ lab boot-resetting start
```

1. Reboot `servera`, and interrupt the countdown in the boot-loader menu.

   1.1. Locate the icon for the `servera` console, as appropriate for your classroom environment. Open the console.

   Send a **`Ctrl`**+**`Alt`**+**`Del`** to your system using the relevant button or menu entry.

   1.2. When the boot-loader menu appears, press any key to interrupt the countdown, except **`Enter`**.

2. Edit the default boot-loader entry, in memory, to abort the boot process just after the kernel mounts all the file systems, but before it hands over control to `systemd`.

   2.1. Use the cursor keys to highlight the default boot-loader entry.

   2.2. Press **`e`** to edit the current entry.

   2.3. Use the cursor keys to navigate to the line that starts with **`linux`**.

   2.4. Press **`End`** to move the cursor to the end of the line.

   2.5. Append **`rd.break`** to the end of the line.

   2.6. Press **`Ctrl`**+**`x`** to boot using the modified configuration.

3. At the **`switch_root`** prompt, remount the **`/sysroot`** file system read/write, then use **`chroot`** to go into a **`chroot`** jail at **`/sysroot`**.

```
switch_root:/# mount -o remount,rw /sysroot
switch_root:/# chroot /sysroot
```

4. Change the `root` password back to `redhat`.

```
sh-4.4# passwd root
Changing password for user root.
New password: redhat
BAD PASSWORD: The password is shorter than 8 characters
Retype new password: redhat
passwd: all authentication tokens updated successfully.
```

5. Configure the system to automatically perform a full SELinux relabel after boot. This is necessary because the **`passwd`** command recreates the **`/etc/shadow`** file without an SELinux context.

```
sh-4.4# touch /.autorelabel
```

6. Type **`exit`** twice to continue booting your system as usual. The system runs an SELinux relabel, then reboots again by itself. When the system is up, verify your work by logging in as `root` at the console. Use `redhat` as the password.

Finish

On `workstation`, run the **`lab boot-resetting finish`** script to complete this exercise.

```
[student@workstation ~]$ lab boot-resetting finish
```

This concludes the guided exercise.

REPAIRING FILE SYSTEM ISSUES AT BOOT

OBJECTIVES

After completing this section, you should be able to manually repair file-system configuration or corruption issues that stop the boot process.

DIAGNOSING AND FIXING FILE SYSTEM ISSUES

Errors in **`/etc/fstab`** and corrupt file systems can stop a system from booting. In most cases, `systemd` drops to an emergency repair shell that requires the `root` password.

The following table lists some common errors and their results.

Common File System Issues

PROBLEM	RESULT
Corrupt file system	`systemd` attempts to repair the file system. If the problem is too severe for an automatic fix, the system drops the user to an emergency shell.
Nonexistent device or UUID referenced in `/etc/fstab`	`systemd` waits for a set amount of time, waiting for the device to become available. If the device does not become available, the system drops the user to an emergency shell after the timeout.
Nonexistent mount point in `/etc/fstab`	The system drops the user to an emergency shell.
Incorrect mount option specified in `/etc/fstab`	The system drops the user to an emergency shell.

In all cases, administrators can also use the emergency target to diagnose and fix the issue, because no file systems are mounted before the emergency shell is displayed.

> **NOTE**
> When using the emergency shell to address file-system issues, do not forget to run **`systemctl daemon-reload`** after editing **`/etc/fstab`**. Without this reload, `systemd` may continue using the old version.

The **`nofail`** option in an entry in the **`/etc/fstab`** file permits the system to boot even if the mount of that file system is not successful. *Do not* use this option under normal circumstances. With **`nofail`**, an application can start with its storage missing, with possibly severe consequences.

REFERENCES

`systemd-fsck`(8), **`systemd-fstab-generator`**(8), and **`systemd.mount`**(5) man pages

GUIDED EXERCISE

REPAIRING FILE SYSTEM ISSUES AT BOOT

In this exercise, you will recover a system from a misconfiguration in **/etc/fstab** that causes the boot process to fail.

OUTCOMES

You should be able to diagnose **/etc/fstab** issues and use emergency mode to recover the system.

BEFORE YOU BEGIN

Log in as the `student` user on `workstation` using `student` as the password.

On `workstation`, run the **lab boot-repairing start** command. This command runs a start script that determines if the `servera` machine is reachable on the network. It also introduces a file-system issue, sets a higher timeout for the GRUB2 menu, and reboots `servera`.

```
[student@workstation ~]$ lab boot-repairing start
```

1. Access the `servera` console and notice that the boot process is stuck early on.

   1.1. Locate the icon for the `servera` console, as appropriate for your classroom environment. Open the console.

   Notice that a start job does not seem to complete. Take a minute to speculate about a possible cause for this behavior.

   1.2. To reboot, send a **Ctrl**+**Alt**+**Del** to your system using the relevant button or menu entry. With this particular boot problem, this key sequence may not immediately abort the running job, and you may have to wait for it to time out before the system reboots.

   If you wait for the task to time out without sending a **Ctrl**+**Alt**+**Del**, the system eventually spawns an emergency shell by itself.

   1.3. When the boot-loader menu appears, press any key to interrupt the countdown, except **Enter**.

2. Looking at the error from the previous boot, it appears that at least parts of the system are still functioning. Because you know the `root` password, `redhat`, attempt an emergency boot.

   2.1. Use the cursor keys to highlight the default boot loader entry.

   2.2. Press **e** to edit the current entry.

   2.3. Use the cursor keys to navigate to the line that starts with **linux**.

2.4. Press **End** to move the cursor to the end of the line.

2.5. Append **`systemd.unit=emergency.target`** to the end of the line.

2.6. Press **`Ctrl`**+**`x`** to boot using the modified configuration.

3. Log in to emergency mode. The `root` password is `redhat`.

```
Give root password for maintenance
(or press Control-D to continue): redhat
[root@servera ~]#
```

4. Determine which file systems are currently mounted.

```
[root@servera ~]# mount
...output omitted...
/dev/vda1 on / type xfs (ro,relatime,seclabel,attr2,inode64,noquota)
...output omitted...
```

Notice that the root file system is mounted read-only.

5. Remount the root file system read/write.

```
[root@servera ~]# mount -o remount,rw /
```

6. Use the **`mount -a`** command to attempt to mount all the other file systems. With the **`--all`** (**`-a`**) option, the command mounts all the file systems listed in **`/etc/fstab`** that are not yet mounted.

```
[root@servera ~]# mount -a
mount: /RemoveMe: mount point does not exist.
```

7. Edit **`/etc/fstab`** to fix the issue.

7.1. Remove or comment out the incorrect line.

```
[root@servera ~]# vim /etc/fstab
...output omitted...
# /dev/sdz1   /RemoveMe   xfs   defaults   0 0
```

7.2. Update `systemd` for the system to register the new **`/etc/fstab`** configuration.

```
[root@servera ~]# systemctl daemon-reload
[root@servera ~]#
```

8. Verify that your **/etc/fstab** is now correct by attempting to mount all entries.

```
[root@servera ~]# mount -a
[root@servera ~]#
```

9. Reboot the system and wait for the boot to complete. The system should now boot normally.

```
[root@servera ~]# systemctl reboot
```

Finish

On `workstation`, run the **lab boot-repairing finish** script to complete this exercise.

```
[student@workstation ~]$ lab boot-repairing finish
```

This concludes the guided exercise.

LAB

CONTROLLING THE BOOT PROCESS

PERFORMANCE CHECKLIST

In this lab, you will reset the `root` password on a system, recover from a misconfiguration, and set the default boot target.

OUTCOMES

You should be able to:

- Reset a lost `root` password.
- Diagnose and fix boot issues.
- Set the default `systemd` target.

BEFORE YOU BEGIN

Log in to `workstation` as `student` using `student` as the password.

On `workstation`, run the **`lab boot-review start`** command. This command runs a start script that determines if the `serverb` machine is reachable on the network. It also introduces a file-system issue, resets the `root` password, sets a higher timeout for the GRUB2 menu, and reboots `serverb`.

```
[student@workstation ~]$ lab boot-review start
```

1. On `serverb`, reset the `root` password to `redhat`.

   Locate the icon for the `serverb` console, as appropriate for your classroom environment. Work from that console.

2. The system fails to boot. A start job does not seem to complete. From the console, fix the issue.

3. Change the default `systemd` target on `serverb` for the system to automatically start a graphical interface when it boots.

   No graphical interface is installed yet on `serverb`. For this exercise, only set the default target and do not install the packages.

Evaluation

On `workstation`, run the **`lab boot-review grade`** script to confirm success on this exercise.

```
[student@workstation ~]$ lab boot-review grade
```

Finish

On `workstation`, run the **`lab boot-review finish`** script to complete the lab.

```
[student@workstation ~]$ lab boot-review finish
```

This concludes the lab.

SOLUTION

CONTROLLING THE BOOT PROCESS

PERFORMANCE CHECKLIST

In this lab, you will reset the `root` password on a system, recover from a misconfiguration, and set the default boot target.

OUTCOMES

You should be able to:

- Reset a lost `root` password.
- Diagnose and fix boot issues.
- Set the default `systemd` target.

BEFORE YOU BEGIN

Log in to `workstation` as `student` using `student` as the password.

On `workstation`, run the **`lab boot-review start`** command. This command runs a start script that determines if the `serverb` machine is reachable on the network. It also introduces a file-system issue, resets the `root` password, sets a higher timeout for the GRUB2 menu, and reboots `serverb`.

```
[student@workstation ~]$ lab boot-review start
```

1. On `serverb`, reset the `root` password to `redhat`.

   Locate the icon for the `serverb` console, as appropriate for your classroom environment. Work from that console.

   1.1. Send a **`Ctrl`**+**`Alt`**+**`Del`** to your system using the relevant button or menu entry.

   1.2. When the boot-loader menu appears, press any key to interrupt the countdown, except **`Enter`**.

   1.3. Use the cursor keys to highlight the default boot loader entry.

   1.4. Press **`e`** to edit the current entry.

   1.5. Use the cursor keys to navigate to the line that starts with **`linux`**.

   1.6. Press **`End`** to move the cursor to the end of the line.

   1.7. Append **`rd.break`** to the end of the line.

   1.8. Press **`Ctrl`**+**`x`** to boot using the modified configuration.

   1.9. At the **`switch_root`** prompt, remount the **`/sysroot`** file system read/write, then use **`chroot`** to go into a **`chroot`** jail at **`/sysroot`**.

```
switch_root:/# mount -o remount,rw /sysroot
switch_root:/# chroot /sysroot
```

1.10. Set the `root` password to `redhat`.

```
sh-4.4# passwd root
Changing password for user root.
New password: redhat
BAD PASSWORD: The password is shorter than 8 characters
Retype new password: redhat
passwd: all authentication tokens updated successfully.
```

1.11. Configure the system to automatically perform a full SELinux relabel after boot.

```
sh-4.4# touch /.autorelabel
```

1.12. Type **`exit`** twice to continue booting your system. The system fails to boot because of an issue you resolve in the next step.

2. The system fails to boot. A start job does not seem to complete. From the console, fix the issue.

   2.1. Boot the system into emergency mode. To do so, reboot `serverb` by sending a **`Ctrl`**+**`Alt`**+**`Del`** to your system using the relevant button or menu entry.

   2.2. When the boot-loader menu appears, press any key to interrupt the countdown, except **`Enter`**.

   2.3. Use the cursor keys to highlight the default boot loader entry.

   2.4. Press **`e`** to edit the current entry.

   2.5. Use the cursor keys to navigate to the line that starts with **`linux`**.

   2.6. Press **`End`** to move the cursor to the end of the line.

   2.7. Append **`systemd.unit=emergency.target`** to the end of the line.

   2.8. Press **`Ctrl`**+**`x`** to boot using the modified configuration.

   2.9. Log in to emergency mode. The `root` password is `redhat`.

```
Give root password for maintenance
(or press Control-D to continue): redhat
[root@serverb ~]#
```

   2.10. Remount the **`/`** file system read/write.

```
[root@serverb ~]# mount -o remount,rw /
```

   2.11. Use the **`mount -a`** command to attempt to mount all the other file systems.

```
[root@serverb ~]# mount -a
mount: /olddata: can't find UUID=4d5c85a5-8921-4a06-8aff-80567e9689bc.
```

   2.12. Edit **`/etc/fstab`** to remove or comment out the incorrect line.

```
[root@serverb ~]# vim /etc/fstab
...output omitted...
#UUID=4d5c85a5-8921-4a06-8aff-80567e9689bc  /olddata  xfs  defaults  0
 0
```

   2.13. Update `systemd` for the system to register the new **`/etc/fstab`** configuration.

```
[root@serverb ~]# systemctl daemon-reload
[root@serverb ~]#
```

   2.14. Verify that your **`/etc/fstab`** is now correct by attempting to mount all entries.

```
[root@serverb ~]# mount -a
[root@serverb ~]#
```

2.15. Reboot the system and wait for the boot to complete. Because you created the **/.autorelabel** file in the first step, after setting the `root` password, the system runs an SELinux relabel, then reboots again by itself. The system should now boot normally.

```
[root@serverb ~]# systemctl reboot
```

3. Change the default `systemd` target on `serverb` for the system to automatically start a graphical interface when it boots.

No graphical interface is installed yet on `serverb`. For this exercise, only set the default target and do not install the packages.

3.1. Log in to `serverb` as the `root` user. Use `redhat` as the password.

3.2. Use the **systemctl set-default** command to set `graphical.target` as the default target.

```
[root@serverb ~]# systemctl set-default graphical.target
```

3.3. Use the **systemctl get-default** command to verify your work.

```
[root@serverb ~]# systemctl get-default
graphical.target
```

3.4. Log off from `serverb`.

```
[root@serverb ~]# exit
```

Evaluation

On `workstation`, run the **lab boot-review grade** script to confirm success on this exercise.

```
[student@workstation ~]$ lab boot-review grade
```

Finish

On `workstation`, run the **lab boot-review finish** script to complete the lab.

```
[student@workstation ~]$ lab boot-review finish
```

This concludes the lab.

SUMMARY

In this chapter, you learned:

- **`systemctl reboot`** and **`systemctl poweroff`** reboot and power down a system, respectively.
- **`systemctl isolate`** ***`target-name`***`.target` switches to a new target at runtime.
- **`systemctl get-default`** and **`systemctl set-default`** can be used to query and set the default target.
- Use **`rd.break`** on the kernel command line to interrupt the boot process before control is handed over from the **`initramfs`**. The root file system is mounted read-only under **`/sysroot`**.
- The emergency target can be used to diagnose and fix file-system issues.

CHAPTER 11

MANAGING NETWORK SECURITY

GOAL

Control network connections to services using the system firewall and SELinux rules.

OBJECTIVES

- Accept or reject network connections to system services using firewalld rules.
- Control whether network services can use specific networking ports by managing SELinux port labels.

SECTIONS

- Managing Server Firewalls (and Guided Exercise)
- Controlling SELinux Port Labeling (and Guided Exercise)

LAB

Managing Server Firewalls

MANAGING SERVER FIREWALLS

OBJECTIVES

After completing this section, you should be able to accept or reject network connections to system services using firewalld rules.

FIREWALL ARCHITECTURE CONCEPTS

The Linux kernel includes `netfilter`, a framework for network traffic operations such as packet filtering, network address translation and port translation. By implementing handlers in the kernel that intercept function calls and messages, `netfilter` allows other kernel modules to interface directly with the kernel's networking stack. Firewall software uses these hooks to register filter rules and packet-modifying functions, allowing every packet going through the network stack to be processed. Any incoming, outgoing, or forwarded network packet can be inspected, modified, dropped, or routed programmatically before reaching user space components or applications. `Netfilter` is the primary component in Red Hat Enterprise Linux 8 firewalls.

Nftables enhances netfilter

The Linux kernel also includes `nftables`, a new filter and packet classification subsystem that has enhanced portions of `netfilter`'s code, but retaining the `netfilter` architecture such as networking stack hooks, connection tracking system, and the logging facility. The advantages of the `nftables` update is faster packet processing, faster ruleset updates, and simultaneous IPv4 and IPv6 processing from the same rules. Another major difference between `nftables` and the original `netfilter` are their interfaces. `Netfilter` is configured through multiple utility frameworks, including `iptables`, `ip6tables`, `arptables`, and `ebtables`, which are now deprecated. Nftables uses the single **`nft`** user-space utility, allowing all protocol management to occur through a single interface, eliminating historical contention caused by diverse front ends and multiple `netfilter` interfaces.

Introducing firewalld

`Firewalld` is a dynamic firewall manager, a front end to the `nftables` framework using the **`nft`** command. Until the introduction of `nftables`, `firewalld` used the **`iptables`** command to configure `netfilter` directly, as an improved alternative to the `iptables` service. In RHEL 8, `firewalld` remains the recommended front end, managing firewall rulesets using **`nft`**. `Firewalld` remains capable of reading and managing `iptables` configuration files and rulesets, using `xtables-nft-multi` to translate `iptables` objects directly into `nftables` rules and objects. Although strongly discouraged, `firewalld` can be configured to revert to the `iptables` back-end for complex use cases where existing `iptables` rulesets cannot be properly processed by **`nft`** translations.

Applications query the subsystem using the `D-Bus` interface. The `firewalld` subsystem, available from the *firewalld* RPM package, is not included in a minimal install, but is included in a base installation. With `firewalld`, firewall management is simplified by classifying all network traffic into *zones*. Based on criteria such as the source IP address of a packet or the incoming network interface, traffic is diverted into the firewall rules for the appropriate zone. Each zone has its own list of ports and services that are either open or closed.

NOTE

For laptops or other machines that regularly change networks, ***NetworkManager*** can be used to automatically set the firewall zone for a connection. The zones are customized with rules appropriate for particular connections.

This is especially useful when traveling between home, work, and public wireless networks. A user might want their system's `sshd` service to be reachable when connected to their home and corporate networks, but not when connected to the public wireless network in the local coffee shop.

Firewalld checks the source address for every packet coming into the system. If that source address is assigned to a specific zone, the rules for that zone apply. If the source address is not assigned to a zone, **`firewalld`** associates the packet with the zone for the incoming network interface and the rules for that zone apply. If the network interface is not associated with a zone for some reason, then **`firewalld`** associates the packet with the default zone.

The default zone is not a separate zone, but is a designation for an existing zone. Initially, **`firewalld`** designates the `public` zone as default, and maps the `lo` loopback interface to the `trusted` zone.

Most zones allow traffic through the firewall, which matches a list of particular ports and protocols, such as **`631/udp`**, or pre-defined services, such as `ssh`. If the traffic does not match a permitted port and protocol or service, it is generally rejected. (The `trusted` zone, which permits all traffic by default, is one exception to this.)

Pre-defined Zones

Firewalld has pre-defined zones, each of which you can customize. By default, all zones permit any incoming traffic which is part of a communication initiated by the system, and all outgoing traffic. The following table details these initial zone configuration.

Default Configuration of Firewalld Zones

ZONE NAME	DEFAULT CONFIGURATION
`trusted`	Allow all incoming traffic.
`home`	Reject incoming traffic unless related to outgoing traffic or matching the `ssh`, `mdns`, `ipp-client`, `samba-client`, or `dhcpv6-client` pre-defined services.
`internal`	Reject incoming traffic unless related to outgoing traffic or matching the `ssh`, `mdns`, `ipp-client`, `samba-client`, or `dhcpv6-client` pre-defined services (same as the `home` zone to start with).
`work`	Reject incoming traffic unless related to outgoing traffic or matching the `ssh`, `ipp-client`, or `dhcpv6-client` pre-defined services.
`public`	Reject incoming traffic unless related to outgoing traffic or matching the `ssh` or `dhcpv6-client` pre-defined services. *The default zone for newly added network interfaces.*

ZONE NAME	DEFAULT CONFIGURATION
`external`	Reject incoming traffic unless related to outgoing traffic or matching the `ssh` pre-defined service. Outgoing IPv4 traffic forwarded through this zone is *masqueraded* to look like it originated from the IPv4 address of the outgoing network interface.
`dmz`	Reject incoming traffic unless related to outgoing traffic or matching the `ssh` pre-defined service.
`block`	Reject all incoming traffic unless related to outgoing traffic.
`drop`	Drop all incoming traffic unless related to outgoing traffic (do not even respond with ICMP errors).

For a list of available pre-defined zones and intended use, see **`firewalld.zones`**(5).

Pre-defined Services

Firewalld has a number of pre-defined services. These service definitions help you identify particular network services to configure. Instead of having to research relevant ports for the `samba-client` service, for example, specify the pre-built `samba-client` service to configure the correct ports and protocols. The following table lists the pre-defined services used in the initial firewall zones configuration.

Selected Pre-defined Firewalld Services

SERVICE NAME	CONFIGURATION
`ssh`	Local SSH server. Traffic to 22/tcp
`dhcpv6-client`	Local DHCPv6 client. Traffic to 546/udp on the fe80::/64 IPv6 network
`ipp-client`	Local IPP printing. Traffic to 631/udp.
`samba-client`	Local Windows file and print sharing client. Traffic to 137/udp and 138/udp.
`mdns`	Multicast DNS (mDNS) local-link name resolution. Traffic to 5353/udp to the 224.0.0.251 (IPv4) or ff02::fb (IPv6) multicast addresses.

NOTE

Many pre-defined services are included in the *firewalld* package. Use **`firewall-cmd --get-services`** to list them. Configuration files for pre-defined services are found in **`/usr/lib/firewalld/services`**, in a format defined by **`firewalld.zone`**(5).

Either use the pre-defined services or directly specify the port and protocol required. The Web Console graphical interface is used to review pre-defined services and to define additional services.

CONFIGURING THE FIREWALL

System administrators interact with `firewalld` in three ways:

- Directly edit configuration files in **`/etc/firewalld/`** (not discussed in this chapter)
- The Web Console graphical interface
- The **`firewall-cmd`** command-line tool

Configuring Firewall Services Using the Web Console

To configure firewall services with Web Console, log in with privileged access by clicking the ***Reuse my password for privileged tasks*** option. This permits the user to execute commands with sudo privileges to modify firewall services.

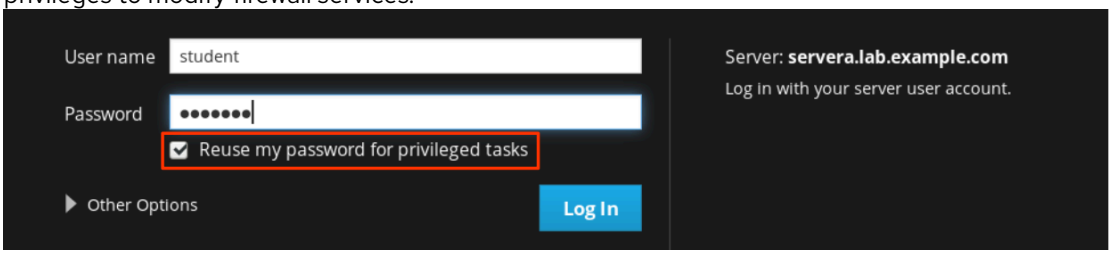

Figure 11.1: The Web Console privileged login

Click the ***Networking*** option in the left navigation menu to display the ***Firewall*** section in the main networking page. Click the ***Firewall*** link to access the allowed services list.

Figure 11.2: The Web Console networking

The allowed services listed are those that are currently permitted by the firewall. Click the arrow (>) to the left of the service name to view service details. To add a service, click the ***Add Services...*** button in the upper right corner of the ***Firewall Allowed Services*** page.

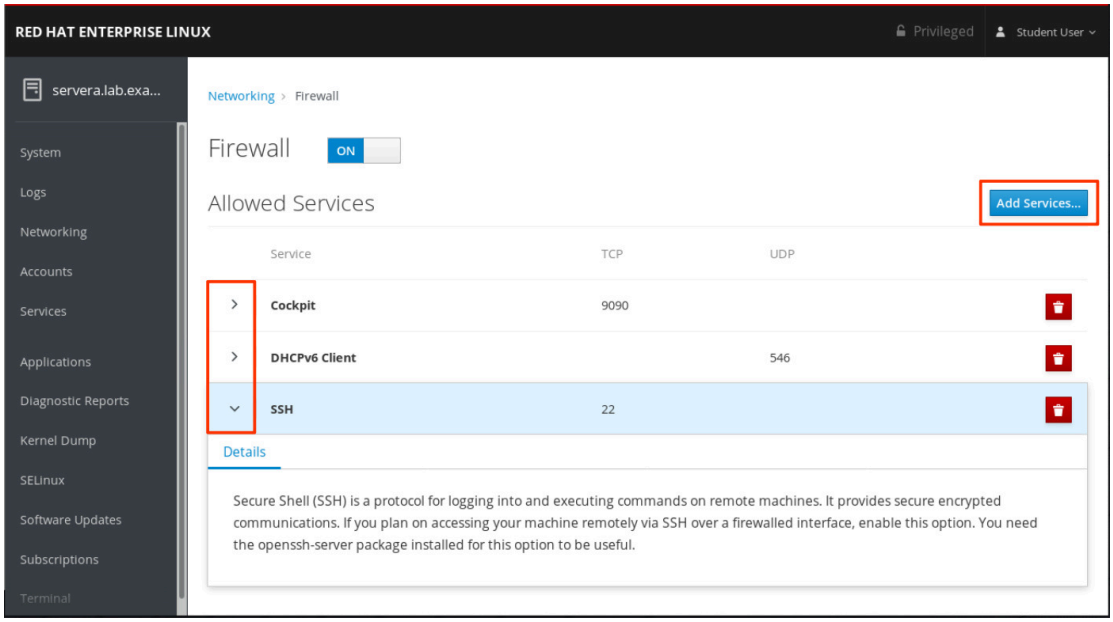

Figure 11.3: The Web Console firewall allowed services list

The ***Add Services*** page displays the available pre-defined services.

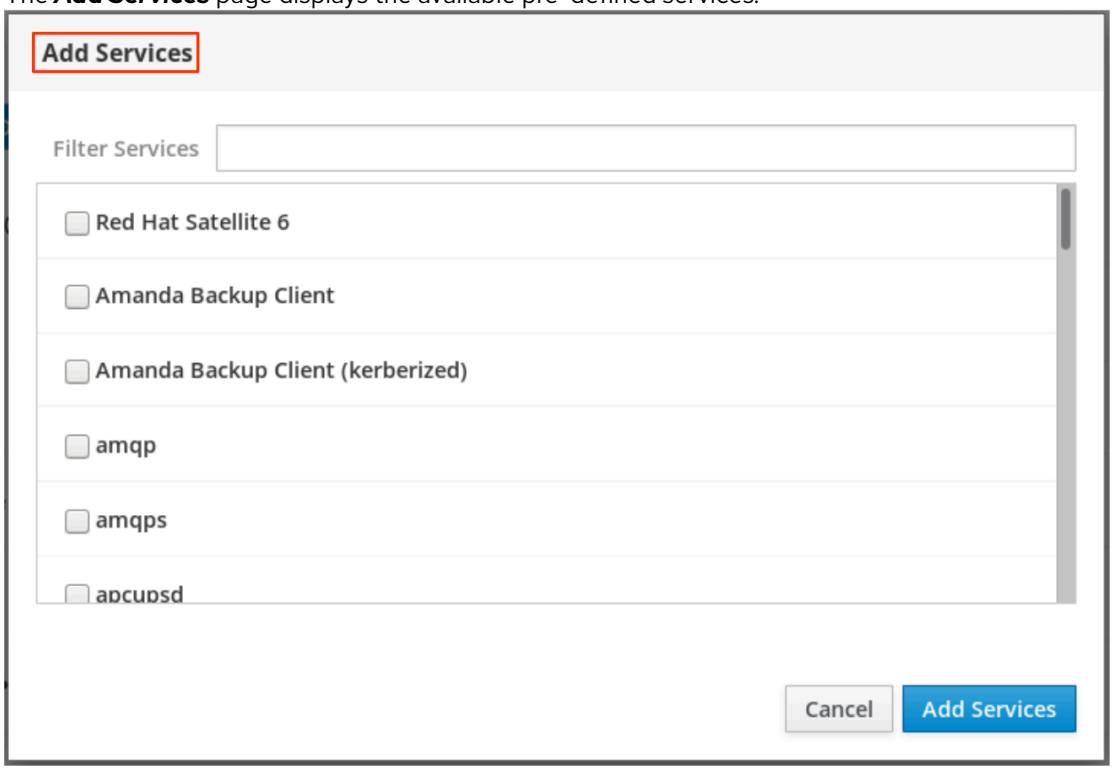

Figure 11.4: The Web Console add services interface

To select a service, scroll through the list or enter a selection in the ***Filter Services*** text box. In the following example, the string **`http`** is entered into the search text box to find services containing that string; that is, web related services. Select the check box to the left of the services to allow through the firewall. Click the ***Add Services*** button to complete the process.

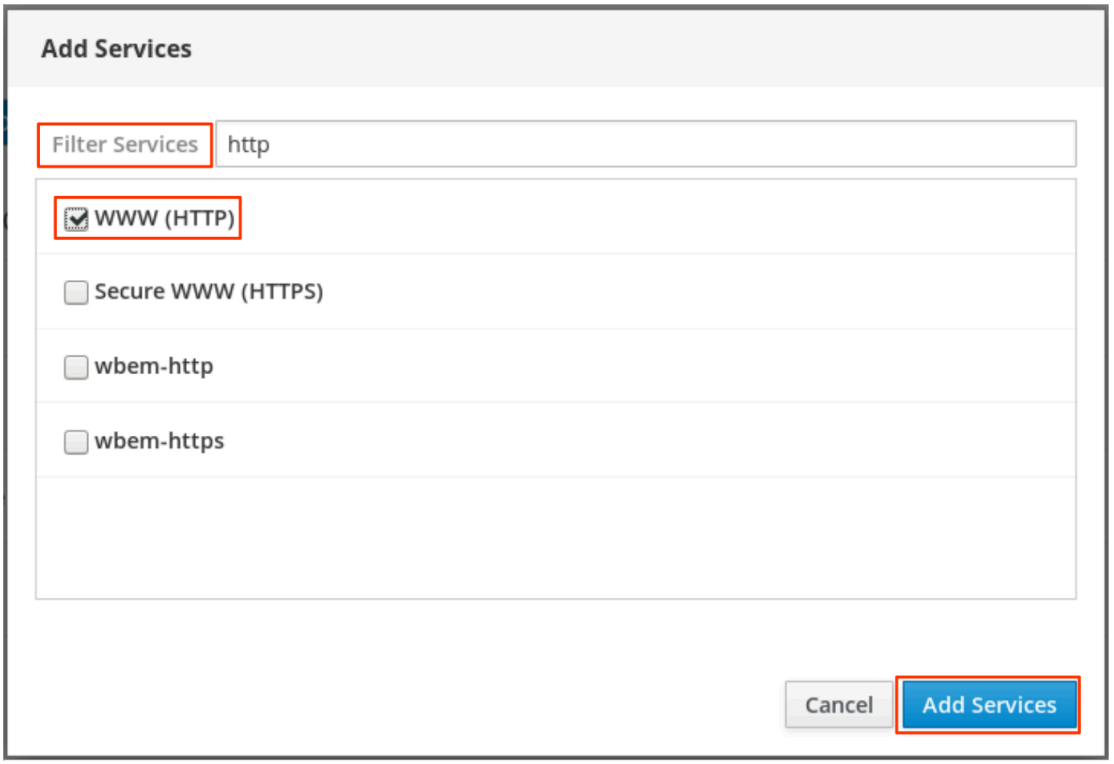

Figure 11.5: The Web Console services filter search

The interface returns to the ***Firewall Allowed Services*** page, where you can review the updated allowed services list.

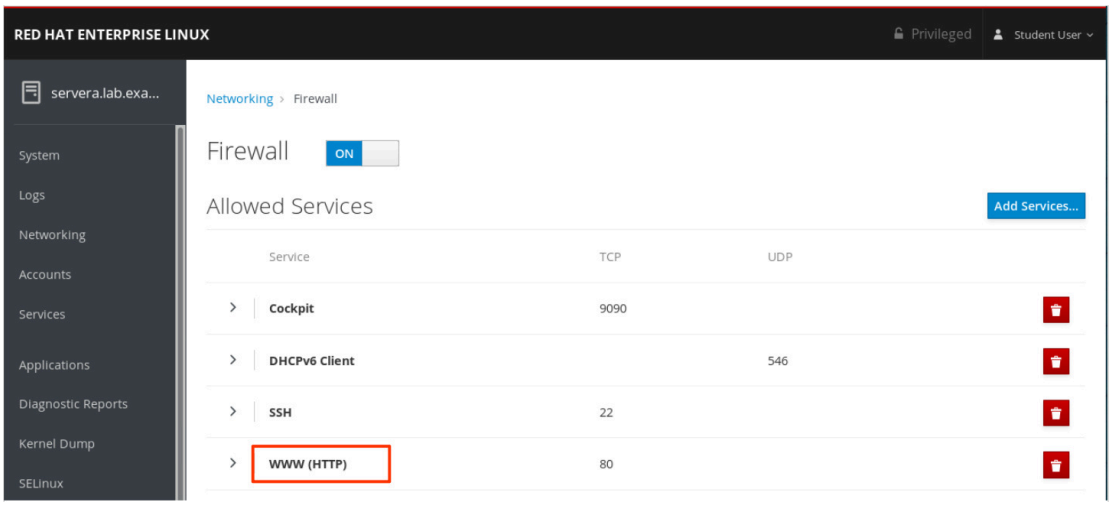

Figure 11.6: The Web Console services list

Configuring the Firewall from the Command Line

The **`firewall-cmd`** command interacts with the **`firewalld`** dynamic firewall manager. It is installed as part of the main *firewalld* package and is available for administrators who prefer to work on the command line, for working on systems without a graphical environment, or for scripting a firewall setup.

The following table lists a number of frequently used **`firewall-cmd`** commands, along with an explanation. Note that unless otherwise specified, almost all commands will work on the *runtime* configuration, unless the **`--permanent`** option is specified. If the **`--permanent`** option

is specified, you must activate the setting by also running the **firewall-cmd --reload** command, which reads the current permanent configuration and applies it as the new runtime configuration. Many of the commands listed take the **--zone=*ZONE*** option to determine which zone they affect. Where a netmask is required, use CIDR notation, such as 192.168.1/24.

FIREWALL-CMD COMMANDS	EXPLANATION
--get-default-zone	Query the current default zone.
--set-default-zone=*ZONE*	Set the default zone. This changes both the runtime and the permanent configuration.
--get-zones	List all available zones.
--get-active-zones	List all zones currently in use (have an interface or source tied to them), along with their interface and source information.
--add-source=*CIDR* [--zone=*ZONE*]	Route all traffic coming from the IP address or network/netmask to the specified zone. If no **--zone=** option is provided, the default zone is used.
--remove-source=*CIDR* [--zone=*ZONE*]	Remove the rule routing all traffic from the zone coming from the IP address or network/netmask network. If no **--zone=** option is provided, the default zone is used.
--add-interface=*INTERFACE* [--zone=*ZONE*]	Route all traffic coming from *INTERFACE* to the specified zone. If no **--zone=** option is provided, the default zone is used.
--change-interface=*INTERFACE* [--zone=*ZONE*]	Associate the interface with *ZONE* instead of its current zone. If no **--zone=** option is provided, the default zone is used.
--list-all [--zone=*ZONE*]	List all configured interfaces, sources, services, and ports for *ZONE*. If no **--zone=** option is provided, the default zone is used.
--list-all-zones	Retrieve all information for all zones (interfaces, sources, ports, services).
--add-service=*SERVICE* [--zone=*ZONE*]	Allow traffic to *SERVICE*. If no **--zone=** option is provided, the default zone is used.
--add-port=*PORT/PROTOCOL* [--zone=*ZONE*]	Allow traffic to the *PORT/PROTOCOL* port(s). If no **--zone=** option is provided, the default zone is used.
--remove-service=*SERVICE* [--zone=*ZONE*]	Remove *SERVICE* from the allowed list for the zone. If no **--zone=** option is provided, the default zone is used.

FIREWALL-CMD COMMANDS	EXPLANATION
--remove-port=*PORT/PROTOCOL* [--zone=*ZONE*]	Remove the *PORT/PROTOCOL* port(s) from the allowed list for the zone. If no **--zone=** option is provided, the default zone is used.
--reload	Drop the runtime configuration and apply the persistent configuration.

The example commands below set the default zone to `dmz`, assign all traffic coming from the `192.168.0.0/24` network to the `internal` zone, and open the network ports for the `mysql` service on the `internal` zone.

```
[root@host ~]# firewall-cmd --set-default-zone=dmz
[root@host ~]# firewall-cmd --permanent --zone=internal \
--add-source=192.168.0.0/24
[root@host ~]# firewall-cmd --permanent --zone=internal --add-service=mysql
[root@host ~]# firewall-cmd --reload
```

NOTE

For situations where the basic syntax of **firewalld** is not enough, you can also add *rich-rules*, a more expressive syntax, to write complex rules. If even the rich-rules syntax is not enough, you can also use *Direct Configuration* rules, raw **nft** syntax mixed in with **firewalld** rules.

These advanced modes are beyond the scope of this chapter.

REFERENCES

firewall-cmd(1), **firewalld**(1), **firewalld.zone**(5), **firewalld.zones**(5), and **nft**(8) man pages

GUIDED EXERCISE

MANAGING SERVER FIREWALLS

In this exercise, you will control access to system services by adjusting system firewall rules with `firewalld`.

OUTCOMES

You should be able to configure firewall rules to control access to services.

BEFORE YOU BEGIN

Log in as the `student` user on `workstation` using `student` as the password.

From `workstation`, run the **`lab netsecurity-firewalls start`** command. The command runs a start script to determine whether the `servera` host is reachable on the network.

```
[student@workstation ~]$ lab netsecurity-firewalls start
```

1. From `workstation`, use SSH to log in to `servera` as `student` user. The systems are configured to use SSH keys for authentication, so a password is not required.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

2. On the `servera` system, ensure that both *httpd* and *mod_ssl* packages are installed. These packages provide the *Apache* web server you will protect with a firewall, and the necessary extensions for the web server to serve content over SSL.

```
[student@servera ~]$ sudo yum install httpd mod_ssl
[sudo] password for student: student
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
```

3. As the `student` user on `servera`, create the **`/var/www/html/index.html`** file. Add one line of text that reads: **`I am servera.`**

```
[student@servera ~]$ sudo bash -c \
"echo 'I am servera.' > /var/www/html/index.html"
```

4. Start and enable the `httpd` service on your `servera` system.

```
[student@servera ~]$ sudo systemctl enable --now httpd
Created symlink /etc/systemd/system/multi-user.target.wants/httpd.service
 → /usr/lib/systemd/system/httpd.service.
```

5. Exit from `servera`.

```
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```

6. From `workstation`, attempt to access your web server on `servera` using both the cleartext port `80/TCP` and the SSL encapsulated port `443/TCP`. Both attempts should fail.

   6.1. This command should fail:

```
[student@workstation ~]$ curl -k http://servera.lab.example.com
curl: (7) Failed to connect to servera.lab.example.com port 80: No
 route to host
```

   6.2. This command should also fail:

```
[student@workstation ~]$ curl -k https://servera.lab.example.com
curl: (7) Failed to connect to servera.lab.example.com port 443: No
 route to host
```

7. Log in to `servera` as the `student` user.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

8. On `servera`, make sure that the `nftables` service is masked and the `firewalld` service is enabled and running.

8.1. Determine whether the status of the `nftables` service is `masked`.

```
[student@servera ~]$ sudo systemctl status nftables
[sudo] password for student: student
● nftables.service - Netfilter Tables
   Loaded: loaded (/usr/lib/systemd/system/nftables.service; disabled;
 vendor preset: disabled)
   Active: inactive (dead)
     Docs: man:nft(8)
```

The results show that `nftables` is disabled and inactive but not masked. Run the following command to mask the service.

```
[student@servera ~]$ sudo systemctl mask nftables
Created symlink /etc/systemd/system/nftables.service → /dev/null.
```

8.2. Verify that the status of the `nftables` service is `masked`.

```
[student@servera ~]$ sudo systemctl status nftables
● nftables.service
   Loaded: masked (Reason: Unit nftables.service is masked.)
   Active: inactive (dead)
```

8.3. Verify that the status of the `firewalld` service is enabled and running.

```
[student@servera ~]$ sudo systemctl status firewalld
● firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; enabled;
 vendor preset: enabled)
   Active: active (running) since Wed 2019-05-22 15:36:02 CDT; 5min
 ago
     Docs: man:firewalld(1)
 Main PID: 703 (firewalld)
    Tasks: 2 (limit: 11405)
   Memory: 29.8M
   CGroup: /system.slice/firewalld.service
```

```
             └─703 /usr/libexec/platform-python -s /usr/sbin/firewalld
 --nofork --nopid

May 22 15:36:01 jegui.ilt.example.com systemd[1]: Starting firewalld -
 dynamic firewall daemon...
May 22 15:36:02 jegui.ilt.example.com systemd[1]: Started firewalld -
 dynamic firewall daemon.
```

8.4. Exit from `servera`.

```
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```

9. From `workstation`, open Firefox and log in to the Web Console running on `servera` to add the `httpd` service to the `public` network zone.

9.1. Open Firefox and browse to `https://servera.lab.example.com:9090` to access the Web Console. Accept the self-signed certificate used by `servera` by adding an exception.

9.2. Select the check box next to ***Reuse my password for privileged tasks*** to ensure administrative privileges.

Log in as `student` user with `student` as the password.

9.3. Click ***Networking*** in the left navigation bar.

9.4. Click the ***Firewall*** link in main ***Networking*** page.

9.5. Click the ***Add Services...*** button located in the upper right side of the ***Firewall*** page.

9.6. In the ***Add Services*** user interface, scroll down or use ***Filter Services*** to locate and select the check box next to the ***Secure WWW (HTTPS)*** service.

9.7. Click the ***Add Services*** button located at the lower right side of the ***Add Services*** user interface.

10. Return to a terminal on `workstation` and verify your work by attempting to view the web server contents of `servera`.

    10.1. This command should fail:

```
[student@workstation ~]$ curl -k http://servera.lab.example.com
curl: (7) Failed to connect to servera.lab.example.com port 80: No
 route to host
```

    10.2. This command should succeed:

```
[student@workstation ~]$ curl -k https://servera.lab.example.com
I am servera.
```

NOTE
If you use Firefox to connect to the web server, it will prompt for verification of the host certificate if it successfully gets past the firewall.

Finish

On `workstation`, run the **`lab netsecurity-firewalls finish`** script to complete this exercise.

```
[student@workstation ~]$ lab netsecurity-firewalls finish
```

This concludes the guided exercise.

CONTROLLING SELINUX PORT LABELING

OBJECTIVES

After completing this section, you should be able to verify that network ports have the correct SELinux type so that services are able to bind to them.

SELINUX PORT LABELING

Figure 11.6: Managing SELinux port security

SELinux does more than just file and process labeling. Network traffic is also tightly enforced by the SELinux policy. One of the methods that SELinux uses for controlling network traffic is labeling network ports; for example, in the **`targeted`** policy, port **`22/TCP`** has the label **`ssh_port_t`** associated with it. The default HTTP ports, **`80/TCP`** and **`443/TCP`**, have the label **`http_port_t`** associated with them.

Whenever a process wants to listen on a port, SELinux checks to see whether the label associated with that process (the domain) is allowed to bind that port label. This can stop a rogue service from taking over ports otherwise used by other (legitimate) network services.

MANAGING SELINUX PORT LABELING

If you decide to run a service on a nonstandard port, SELinux almost certainly will block the traffic. In this case, you must update SELinux port labels. In some cases, the **`targeted`** policy has already labeled the port with a type that can be used; for example, since port **`8008/TCP`** is often used for web applications, that port is already labeled with **`http_port_t`**, the default port type for the web server.

Listing Port Labels

To get an overview of all the current port label assignments, run the **`semanage port -l`** command. The **`-l`** option lists all current assignments in this form:

```
port_label_t     tcp|udp    comma,separated,list,of,ports
```

Example output:

```
[root@host ~]# semanage port -l
...output omitted...
http_cache_port_t       tcp   8080, 8118, 8123, 10001-10010
http_cache_port_t       udp   3130
http_port_t             tcp   80, 81, 443, 488, 8008, 8009, 8443, 9000
...output omitted...
```

To refine the search, use the **`grep`** command:

```
[root@host ~]# semanage port -l | grep ftp
ftp_data_port_t                tcp      20
ftp_port_t                     tcp      21, 989, 990
ftp_port_t                     udp      989, 990
tftp_port_t                    udp      69
```

Note that a port label can appear twice in the output, once for TCP and once for UDP.

Managing Port Labels

Use the **semanage** command to assign new port labels, remove port labels, or modify existing ones.

IMPORTANT

Most standard services available in the Linux distribution provide an SELinux policy module that sets labels on ports. You cannot change the labels on those ports using **semanage**; to change those, you need to replace the policy module. Writing and generating policy modules falls outside the scope of this course.

To add a port to an existing port label (type), use the following syntax. The **-a** adds a new port label, the **-t** denotes the type, the **-p** denotes the protocol.

```
[root@host ~]# semanage port -a -t port_label -p tcp|udp PORTNUMBER
```

For example, to allow a `gopher` service to listen on port **71/TCP**:

```
[root@host~]# semanage port -a -t gopher_port_t -p tcp 71
```

To view local changes to the default policy, administrators can add the **-C** option to the **semanage** command.

```
[root@host~]# semanage port -l -C
SELinux Port Type              Proto    Port Number

gopher_port_t                  tcp      71
```

NOTE

The **targeted** policy ships with a large number of port types.

Service specific SELinux man pages found in the *selinux-policy-doc* package include documentation on SELinux types, booleans, and port types. If these man pages are not yet installed on your system, follow this procedure:

```
[root@host ~]# yum -y install selinux-policy-doc
[root@host ~]# man -k _selinux
```

Removing Port Labels

The syntax for removing a custom port label is the same as the syntax for adding a port label, but instead of using the **-a** option (for Add), use the **-d** option (for Delete).

For example, to remove the binding of port **71/TCP** to **gopher_port_t**:

```
[root@host ~]# semanage port -d -t gopher_port_t -p tcp 71
```

Modifying Port Bindings

To change a port binding, perhaps because requirements changed, use the **-m** (Modify) option. This is a more efficient process than removing the old binding and adding a new one.

For example, to modify port **71/TCP** from **gopher_port_t** to **http_port_t**, an administrator can use the following command:

```
[root@server ~]# semanage port -m -t http_port_t -p tcp 71
```

As before, view the modification using the **semanage** command.

```
[root@server ~]# semanage port -l -C
SELinux Port Type              Proto    Port Number

http_port_t                    tcp      71
[root@server ~]# semanage port -l | grep http
http_cache_port_t              tcp      8080, 8118, 8123, 10001-10010
http_cache_port_t              udp      3130
http_port_t                    tcp      71, 80, 81, 443, 488, 8008, 8009, 8443,
 9000
pegasus_http_port_t            tcp      5988
pegasus_https_port_t           tcp      5989
```

REFERENCES

semanage(8), **semanage-port**(8), and ***_selinux**(8) man pages

GUIDED EXERCISE

CONTROLLING SELINUX PORT LABELING

In this lab, you will configure your `servera` system to allow HTTP access on a nonstandard port.

OUTCOMES:

You will configure a web server running on `servera` successfully serving content using a nonstandard port.

BEFORE YOU BEGIN

Log in as the `student` user on `workstation` using `student` as the password.

On `workstation`, run the **`lab netsecurity-ports start`** command. This command runs a start script that determines whether the `servera` machine is reachable on the network. It also installs the `httpd` service and configures the firewall on `servera` to allow http connections.

```
[student@workstation ~]$ lab netsecurity-ports start
```

Your organization is deploying a new custom web application. The web application is running on a nonstandard port; in this case, **`82/TCP`**.

One of your junior administrators has already configured the application on your `servera`. However, the web server content is not accessible.

1. Use the **`ssh`** command to log in to `servera` as the `student` user. The systems are configured to use SSH keys for authentication, so a password is not required.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

2. Use the **`sudo -i`** command to switch to the `root` user. The password for the `student` user is `student`.

```
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

3. Attempt to fix the web content problem by restarting the `httpd` service.

   3.1. Use the **`systemctl`** command to restart the **`httpd.service`**. This command is expected to fail.

```
[root@servera ~]# systemctl restart httpd.service
Job for httpd.service failed because the control process exited with
 error code.
See "systemctl status httpd.service" and "journalctl -xe" for details.
```

3.2. Use the **`systemctl status -l`** command to reveal the status of the `httpd` service. Note the **`permission denied`** error.

```
[root@servera ~]# systemctl status -l httpd.service
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled;
 vendor preset: disabled)
   Active: failed (Result: exit-code) since Mon 2019-04-08 14:23:29
 CEST; 3min 33s ago
    Docs: man:httpd.service(8)
   Process: 28078 ExecStart=/usr/sbin/httpd $OPTIONS -DFOREGROUND
 (code=exited, status=1/FAILURE)
 Main PID: 28078 (code=exited, status=1/FAILURE)
   Status: "Reading configuration..."

Apr 08 14:23:29 servera.lab.example.com systemd[1]: Starting The
 Apache HTTP Server...
Apr 08 14:23:29 servera.lab.example.com httpd[28078]: (13)Permission
 denied: AH00072: make_sock: could not bind to address [::]:82
Apr 08 14:23:29 servera.lab.example.com httpd[28078]: (13)Permission
 denied: AH00072: make_sock: could not bind to address 0.0.0.0:82
Apr 08 14:23:29 servera.lab.example.com httpd[28078]: no listening
 sockets available, shutting down
Apr 08 14:23:29 servera.lab.example.com httpd[28078]: AH00015: Unable
 to open logs
Apr 08 14:23:29 servera.lab.example.com systemd[1]: httpd.service:
 Main process exited, code=exited, status=1/FAILURE
Apr 08 14:23:29 servera.lab.example.com systemd[1]: httpd.service:
 Failed with result 'exit-code'.
```

3.3. Use the **`sealert`** command to check if SELinux is blocking **`httpd`** from binding to port **`82/TCP`**.

```
[root@servera ~]# sudo sealert -a /var/log/audit/audit.log
100% done
found 1 alerts in /var/log/audit/audit.log
--------------------------------------------------------------------------------

SELinux is preventing /usr/sbin/httpd from name_bind access on the
 tcp_socket port 82.

*****  Plugin bind_ports (99.5 confidence) suggests
 ************************

If you want to allow /usr/sbin/httpd to bind to network port 82
Then you need to modify the port type.
Do
```

```
# semanage port -a -t PORT_TYPE -p tcp 82
    where PORT_TYPE is one of the following: http_cache_port_t,
 http_port_t, jboss_management_port_t, jboss_messaging_port_t,
 ntop_port_t, puppet_port_t.
...output omitted...
Raw Audit Messages
type=AVC msg=audit(1554726569.188:852): avc:
 denied  { name_bind } for  pid=28393 comm="httpd"
 src=82 scontext=system_u:system_r:httpd_t:s0
 tcontext=system_u:object_r:reserved_port_t:s0 tclass=tcp_socket
 permissive=0
```

4. Configure SELinux to allow **`httpd`** to bind to port **`82/TCP`**, then restart the **`httpd.service`** service.

   4.1. Use the **`semanage`** command to find an appropriate port type for port **`82/TCP`**.

```
[root@servera ~]# semanage port -l | grep http
http_cache_port_t              tcp      8080, 8118, 8123, 10001-10010
http_cache_port_t              udp      3130
http_port_t                    tcp      80, 81, 443, 488, 8008, 8009,
 8443, 9000
pegasus_http_port_t            tcp      5988
pegasus_https_port_t           tcp      5989
```

   `http_port_t` contains the default HTTP ports, **`80/TCP`** and **`443/TCP`**. This is the correct port type for the web server.

   4.2. Use the **`semanage`** command to assign port **`82/TCP`** the **`http_port_t`** type.

```
[root@servera ~]# semanage port -a -t http_port_t -p tcp 82
```

   4.3. Use the **`systemctl`** command to restart the **`httpd.service`** service. This command should succeed.

```
[root@servera ~]# systemctl restart httpd.service
```

5. Check if you can now access the web server running on port **`82/TCP`**. Use the **`curl`** command to access the web service from `servera`.

```
[root@servera ~]# curl http://servera.lab.example.com:82
Hello
```

6. In a different terminal window, check whether you can access the new web service from `workstation`. Use the **`curl`** command to access the web service from `workstation`.

```
[student@workstation ~]$ curl http://servera.lab.example.com:82
curl: (7) Failed to connect to servera.example.com:82; No route to host
```

That error means you still can not connect to the web service from `workstation`.

7. On `servera`, open up port **`82/TCP`** on the firewall.

   7.1. Use the **`firewall-cmd`** command to open port **`82/TCP`** in the permanent configuration for the default zone on the firewall on `servera`.

```
[root@servera ~]# firewall-cmd --permanent --add-port=82/tcp
success
```

   7.2. Activate your firewall changes on `servera`.

```
[root@servera ~]# firewall-cmd --reload
success
```

8. Use the **`curl`** command to access the web service from `workstation`.

```
[student@workstation ~]$ curl http://servera.lab.example.com:82
Hello
```

9. Exit from `servera`.

```
[root@servera ~]# exit
logout
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```

Finish

On `workstation`, run the **`lab netsecurity-ports finish`** script to complete this exercise.

```
[student@workstation ~]$ lab netsecurity-ports finish
```

This concludes the guided exercise.

LAB

MANAGING NETWORK SECURITY

PERFORMANCE CHECKLIST

In this lab, you will configure firewall and SELinux settings to allow access to multiple web servers running on `serverb`.

OUTCOMES

You should be able to configure firewall and SELinux settings on a web server host.

BEFORE YOU BEGIN

Log in as the `student` user on `workstation` using `student` as the password.

From `workstation`, run the **`lab netsecurity-review start`** command. The command runs a start script to determine whether the `serverb` host is reachable on the network.

```
[student@workstation ~]$ lab netsecurity-review start
```

Your company has decided to run a new web app. This application listens on ports **`80/TCP`** and **`1001/TCP`**. Port **`22/TCP`** for **`ssh`** access must also be available. All changes you make should persist across a reboot.

If prompted by **`sudo`**, use `student` as the password.

Important: The graphical interface used in the Red Hat Online Learning environment needs port **`5900/TCP`** to remain available as well. This port is also known under the service name `vnc-server`. If you accidentally lock yourself out from your `serverb`, you can either attempt to recover by using **`ssh`** to your `serverb` machine from your `workstation` machine, or reset your `serverb` machine. If you elect to reset your `serverb` machine, you must run the setup scripts for this lab again. The configuration on your machines already includes a custom zone called `ROL` that opens these ports.

1. From `workstation`, test access to the default web server at `http://serverb.lab.example.com` and to the virtual host at `http://serverb.lab.example.com:1001`.
2. Log in to `serverb` to determine what is preventing access to the web servers.
3. Configure SELinux to allow the `httpd` service to listen on port **`1001/TCP`**.
4. From `workstation`, test access to the default web server at `http://serverb.lab.example.com` and to the virtual host at `http://serverb.lab.example.com:1001`.
5. Log in to `serverb` to determine whether the correct ports are assigned to the firewall.
6. Add port **`1001/TCP`** to the permanent configuration for the `public` network zone. Confirm your configuration.

7. From `workstation`, confirm that the default web server at `serverb.lab.example.com` returns **`SERVER B`** and the virtual host at `serverb.lab.example.com:1001` returns **`VHOST 1`**.

Evaluation

On `workstation`, run the **`lab netsecurity-review grade`** command to confirm success of this lab exercise.

```
[student@workstation ~]$ lab netsecurity-review grade
```

Finish

On `workstation`, run the **`lab netsecurity-review finish`** script to complete this exercise.

```
[student@workstation ~]$ lab netsecurity-review finish
```

This concludes the lab.

SOLUTION

MANAGING NETWORK SECURITY

PERFORMANCE CHECKLIST

In this lab, you will configure firewall and SELinux settings to allow access to multiple web servers running on `serverb`.

OUTCOMES

You should be able to configure firewall and SELinux settings on a web server host.

BEFORE YOU BEGIN

Log in as the `student` user on `workstation` using `student` as the password.

From `workstation`, run the **`lab netsecurity-review start`** command. The command runs a start script to determine whether the `serverb` host is reachable on the network.

```
[student@workstation ~]$ lab netsecurity-review start
```

Your company has decided to run a new web app. This application listens on ports **`80/TCP`** and **`1001/TCP`**. Port **`22/TCP`** for **`ssh`** access must also be available. All changes you make should persist across a reboot.

If prompted by **`sudo`**, use `student` as the password.

Important: The graphical interface used in the Red Hat Online Learning environment needs port **`5900/TCP`** to remain available as well. This port is also known under the service name `vnc-server`. If you accidentally lock yourself out from your `serverb`, you can either attempt to recover by using **`ssh`** to your `serverb` machine from your `workstation` machine, or reset your `serverb` machine. If you elect to reset your `serverb` machine, you must run the setup scripts for this lab again. The configuration on your machines already includes a custom zone called `ROL` that opens these ports.

1. From `workstation`, test access to the default web server at `http://serverb.lab.example.com` and to the virtual host at `http://serverb.lab.example.com:1001`.

   1.1. Test access to the `http://serverb.lab.example.com` web server. The test currently fails. Ultimately, the web server should return **`SERVER B`**.

   ```
   [student@workstation ~]$ curl http://serverb.lab.example.com
   curl: (7) Failed to connect to serverb.lab.example.com port 80:
    Connection refused
   ```

   1.2. Test access to the `http://serverb.lab.example.com:1001` virtual host. The test currently fails. Ultimately, the virtual host should return **`VHOST 1`**.

```
[student@workstation ~]$ curl http://serverb.lab.example.com:1001
curl: (7) Failed to connect to serverb.lab.example.com port 1001: No
 route to host
```

2. Log in to `serverb` to determine what is preventing access to the web servers.

2.1. From `workstation`, open an SSH session to `serverb` as `student` user. The systems are configured to use SSH keys for authentication, so a password is not required.

```
[student@workstation ~]$ ssh student@serverb
...output omitted...
[student@serverb ~]$
```

2.2. Determine whether the `httpd` service is active.

```
[student@serverb ~]$ systemctl is-active httpd
inactive
```

2.3. Enable and start the `httpd` service. The `httpd` service fails to start.

```
[student@serverb ~]$ sudo systemctl enable --now httpd
[sudo] password for student: student
Created symlink /etc/systemd/system/multi-user.target.wants/
httpd.service → /usr/lib/systemd/system/httpd.service.
Job for httpd.service failed because the control process exited with
 error code.
See "systemctl status httpd.service" and "journalctl -xe" for details.
```

2.4. Investigate the reasons why the `httpd.service` service failed to start.

```
[student@serverb ~]$ systemctl status httpd.service
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled;
 vendor preset: disabled)
   Active: failed (Result: exit-code) since Thu 2019-04-11 19:25:36
 CDT; 19s ago
     Docs: man:httpd.service(8)
  Process: 9615 ExecStart=/usr/sbin/httpd $OPTIONS -DFOREGROUND
 (code=exited, status=1/FAILURE)
 Main PID: 9615 (code=exited, status=1/FAILURE)
   Status: "Reading configuration..."

Apr 11 19:25:36 serverb.lab.example.com systemd[1]: Starting The Apache
 HTTP Server...
Apr 11 19:25:36 serverb.lab.example.com httpd[9615]: (13)Permission
 denied: AH00072: make_sock: could not bind to address [::]:1001
Apr 11 19:25:36 serverb.lab.example.com httpd[9615]: (13)Permission
 denied: AH00072: make_sock: could not bind to address 0.0.0.0:1001
Apr 11 19:25:36 serverb.lab.example.com httpd[9615]: no listening
 sockets available, shutting down
```

```
Apr 11 19:25:36 serverb.lab.example.com httpd[9615]: AH00015: Unable to
 open logs
Apr 11 19:25:36 serverb.lab.example.com systemd[1]: httpd.service: Main
 process exited, code=exited, status=1/FAILURE
Apr 11 19:25:36 serverb.lab.example.com systemd[1]: httpd.service:
 Failed with result 'exit-code'.
Apr 11 19:25:36 serverb.lab.example.com systemd[1]: Failed to start The
 Apache HTTP Server.
```

2.5. Use the **sealert** command to check whether SELinux is blocking the `httpd` service from binding to port **1001/TCP**.

```
[student@serverb ~]$ sudo sealert -a /var/log/audit/audit.log
100% done
found 1 alerts in /var/log/audit/audit.log
--------------------------------------------------------------------------------

SELinux is preventing /usr/sbin/httpd from name_bind access on the
 tcp_socket port 1001.

*****  Plugin bind_ports (99.5 confidence) suggests
 ************************

If you want to allow /usr/sbin/httpd to bind to network port 1001
Then you need to modify the port type.
Do
# semanage port -a -t PORT_TYPE -p tcp 1001
    where PORT_TYPE is one of the following: http_cache_port_t,
 http_port_t, jboss_management_port_t, jboss_messaging_port_t,
 ntop_port_t, puppet_port_t.

*****  Plugin catchall (1.49 confidence) suggests
 **************************

...output omitted...
```

3. Configure SELinux to allow the `httpd` service to listen on port **`1001/TCP`**.

3.1. Use the **`semanage`** command to find the correct port type.

```
[student@serverb ~]$ sudo semanage port -l | grep 'http'
http_cache_port_t      tcp  8080, 8118, 8123, 10001-10010
http_cache_port_t      udp  3130
http_port_t            tcp  80, 81, 443, 488, 8008, 8009, 8443, 9000
pegasus_http_port_t    tcp  5988
pegasus_https_port_t   tcp  5989
```

3.2. Use the **`semanage`** command to bind port **`1001/TCP`** to the `http_port_t` type.

```
[student@serverb ~]$ sudo semanage port -a -t http_port_t -p tcp 1001
[student@serverb ~]$
```

3.3. Confirm that port **`1001/TCP`** is bound to the `http_port_t` port type.

```
[student@serverb ~]$ sudo semanage port -l | grep '^http_port_t'
http_port_t            tcp  1001, 80, 81, 443, 488, 8008, 8009, 8443,
 9000
```

3.4. Enable and start the `httpd` service.

```
[student@serverb ~]$ sudo systemctl enable --now httpd
```

3.5. Verify the running state of the `httpd` service.

```
[student@serverb ~]$ systemctl is-active httpd; systemctl is-enabled
 httpd
active
enabled
```

3.6. Exit from `serverb`.

```
[student@serverb ~]$ exit
logout
Connection to serverb closed.
[student@workstation ~]$
```

4. From `workstation`, test access to the default web server at `http://serverb.lab.example.com` and to the virtual host at `http://serverb.lab.example.com:1001`.

4.1. Test access to the `http://serverb.lab.example.com` web server. The web server should return **SERVER B**.

```
[student@workstation ~]$ curl http://serverb.lab.example.com
SERVER B
```

4.2. Test access to the `http://serverb.lab.example.com:1001` virtual host. The test continues to fail.

```
[student@workstation ~]$ curl http://serverb.lab.example.com:1001
curl: (7) Failed to connect to serverb.lab.example.com port 1001: No
 route to host
```

5. Log in to `serverb` to determine whether the correct ports are assigned to the firewall.

5.1. From `workstation`, log in to `serverb` as the `student` user.

```
[student@workstation ~]$ ssh student@serverb
...output omitted...
[student@serverb ~]$
```

5.2. Verify that the default firewall zone is set to **`public`**.

```
[student@serverb ~]$ firewall-cmd --get-default-zone
public
```

5.3. If the previous step did not return `public` as the default zone, correct it with the following command:

```
[student@serverb ~]$ sudo firewall-cmd --set-default-zone public
```

5.4. Determine the open ports listed in the `public` network zone.

```
[student@serverb ~]$ sudo firewall-cmd --permanent --zone=public --
list-all
[sudo] password for student: student
public
  target: default
  icmp-block-inversion: no
  interfaces:
  sources:
  services: cockpit dhcpv6-client http ssh
  ports:
  protocols:
  masquerade: no
```

```
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

6. Add port **1001/TCP** to the permanent configuration for the `public` network zone. Confirm your configuration.

   6.1. Add port **1001/TCP** to the `public` network zone.

```
[student@serverb ~]$ sudo firewall-cmd --permanent --zone=public \
--add-port=1001/tcp
success
```

   6.2. Reload the firewall configuration.

```
[student@serverb ~]$ sudo firewall-cmd --reload
success
```

   6.3. Confirm your configuration.

```
[student@serverb ~]$ sudo firewall-cmd --permanent --zone=public --
list-all
public
  target: default
  icmp-block-inversion: no
  interfaces:
  sources:
  services: cockpit dhcpv6-client http ssh
  ports: 1001/tcp
  protocols:
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

   6.4. Exit from `serverb`.

```
[student@serverb ~]$ exit
logout
Connection to serverb closed.
[student@workstation ~]$
```

7. From `workstation`, confirm that the default web server at `serverb.lab.example.com` returns **`SERVER B`** and the virtual host at `serverb.lab.example.com:1001` returns **`VHOST 1`**.

   7.1. Test access to the `http://serverb.lab.example.com` web server.

```
[student@workstation ~]$ curl http://serverb.lab.example.com
SERVER B
```

   7.2. Test access to the `http://serverb.lab.example.com:1001` virtual host.

```
[student@workstation ~]$ curl http://serverb.lab.example.com:1001
VHOST 1
```

Evaluation

On `workstation`, run the **`lab netsecurity-review grade`** command to confirm success of this lab exercise.

```
[student@workstation ~]$ lab netsecurity-review grade
```

Finish

On `workstation`, run the **`lab netsecurity-review finish`** script to complete this exercise.

```
[student@workstation ~]$ lab netsecurity-review finish
```

This concludes the lab.

SUMMARY

In this chapter, you learned:

- The `netfilter` subsystem allows kernel modules to inspect every packet traversing the system. All incoming, outgoing or forwarded network packets are inspected.
- The use of **`firewalld`** has simplified management by classifying all network traffic into zones. Each zone has its own list of ports and services. The **`public`** zone is set as the default zone.
- The `firewalld` service ships with a number of pre-defined services. They can be listed using the **`firewall-cmd --get-services`** command.
- Network traffic is tightly controlled by the SELinux policy. Network ports are labeled. For example, port **`22/TCP`** has the label **`ssh_port_t`** associated with it. When a process wants to listen on a port, SELinux checks to see whether the label associated with it is allowed to bind that port label.
- The **`semanage`** command is used to add, delete, and modify labels.

CHAPTER 12

INSTALLING RED HAT ENTERPRISE LINUX

GOAL

Install Red Hat Enterprise Linux on servers and virtual machines.

OBJECTIVES

- Install Red Hat Enterprise Linux on a server.
- Automate the installation process using Kickstart.
- Install a virtual machine on your Red Hat Enterprise Linux server using Cockpit.

SECTIONS

- Installing Red Hat Enterprise Linux (and Guided Exercise)
- Automating Installation with Kickstart (and Guided Exercise)
- Installing and Configuring Virtual Machines (and Quiz)

LAB

Installing Red Hat Enterprise Linux

INSTALLING RED HAT ENTERPRISE LINUX

OBJECTIVES

After completing this section, you should be able to install Red Hat Enterprise Linux on a server.

SELECTING INSTALLATION MEDIA

Red Hat provides several installation media that you can download from the Customer Portal website using your active subscription.

- A binary DVD containing *Anaconda*, the Red Hat Enterprise Linux installation program, and the BaseOS and AppStream package repositories. These repositories contain the packages needed to complete the installation without additional material.
- A boot ISO containing Anaconda, but requires a configured network to access package repositories made available using HTTP, FTP, or NFS.
- A QCOW2 image containing a prebuilt system disk ready to deploy as a virtual machine in cloud or enterprise virtual environments. *QCOW2 (QEMU Copy On Write)* is the standard image format used by Red Hat.

Red Hat provides installation media for four supported processor architectures: x86 64-bit (AMD and Intel), IBM Power Systems (Little Endian), IBM Z, and ARM 64-bit.

After downloading, burn the DVD or boot ISO to physical media, copy each to a USB flash drive or similar, or publish each from a network server for automated Kickstart use.

Building Images with Composer

Composer is a new tool available in RHEL 8. For specialized use cases, Composer allows administrators to build custom system images for deployment on cloud platforms or virtual environments.

Composer uses the Cockpit graphical web console. It can also be invoked from a command line using the **`composer-cli`** command.

MANUAL INSTALLATION WITH ANACONDA

Using the binary DVD or boot ISO, administrators can install a new RHEL system on a bare-metal server or a virtual machine. The Anaconda program supports two installation methods:

- The manual installation interacts with the user to query how Anaconda should install and configure the system.
- The automated installation uses a *Kickstart* file which tells Anaconda how to install the system. A later section discusses Kickstart installations in greater detail.

Installing RHEL with the Graphical Interface

When you boot the system from the binary DVD or the boot ISO, Anaconda starts as a graphical application.

At the ***Welcome to Red Hat Enterprise Linux 8*** screen, select the language to use during installation. This also sets the default language of the system after installation. Individual users can select their own account's preferred language after installation.

Anaconda presents the ***Installation Summary*** window, the central place to customize parameters before beginning the installation.

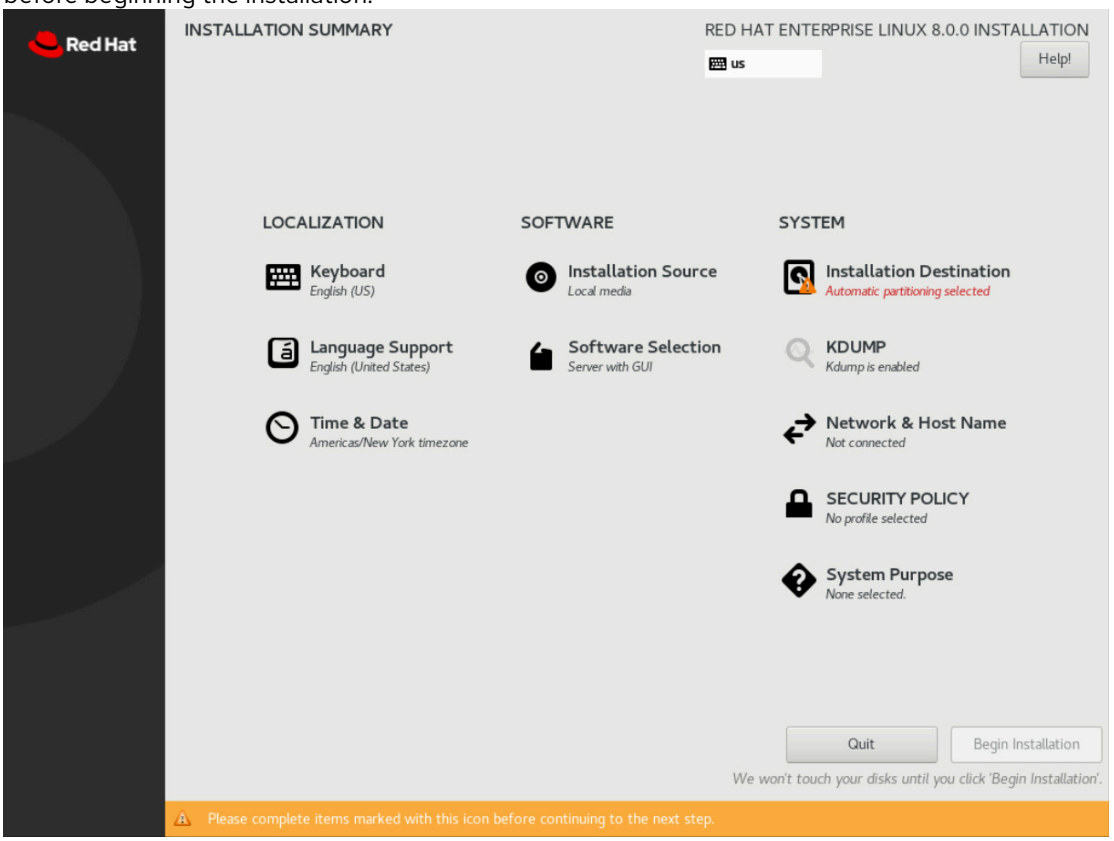

Figure 12.1: Installation Summary window

From this window, configure the installation parameters by selecting the icons in any order. Select an item to view or edit. In any item, click ***Done*** to return to this central screen.

Anaconda marks mandatory items with a triangle warning symbol and message. The orange status bar at the bottom of the screen reminds you that mandatory items remain to be completed before the installation can begin.

Complete the following items as needed:

- ***Keyboard*** - Add additional keyboard layouts.
- ***Language Support*** - Select additional languages to install.
- ***Time & Date*** - Select the system's location city by clicking on the interactive map, or select it from the drop-down list. Specify the local time zone even when using *Network Time Protocol (NTP)*.
- ***Installation Source*** - Provide the source package location that Anaconda needs for installation. If using the binary DVD, the installation source field already refers to the DVD.
- ***Software Selection*** - Select the base environment to install, plus any additional add-ons. The ***Minimal Install*** environment installs only the essential packages to run Red Hat Enterprise Linux.

- ***Installation Destination*** - Select and partition the disks onto which Red Hat Enterprise Linux will install. This item expects an administrator to comprehend partitioning schemes and file system selection criteria. The default radio button for automatic partitioning allocates the selected storage devices using all available space.
- ***KDUMP*** - Kdump is a kernel feature that collects system memory contents when the kernel crashes. Red Hat engineers can analyze a kdump to identify the cause of a crash. Use this Anaconda item to enable or disable Kdump.
- ***Network & Host Name*** - Detected network connections list on the left. Select a connection to display its details. To configure the selected network connection, click ***Configure***.
- ***SECURITY POLICY*** - By activating a security policy profile, such as the *Payment Card Industry Data Security Standard (PCI DSS)* profile, Anaconda applies restrictions and recommendations, defined by the selected profile, during installation.
- ***System Purpose*** - A new installation feature that allocates active system entitlements to match the intended system use.

After completing the installation configuration, and resolving all warnings, click ***Begin Installation***. Clicking ***Quit*** aborts the installation without applying any changes to the system.

While the system is installing, complete the following items when they display:

- ***Root Password*** - The installation program prompts to set a `root` password. The final stage of the installation process will not continue until you define a `root` password.
- ***User Creation*** - Create an optional non-root account. Maintaining a local, general use account is a recommended practice. Accounts can also be created after the installation is complete.

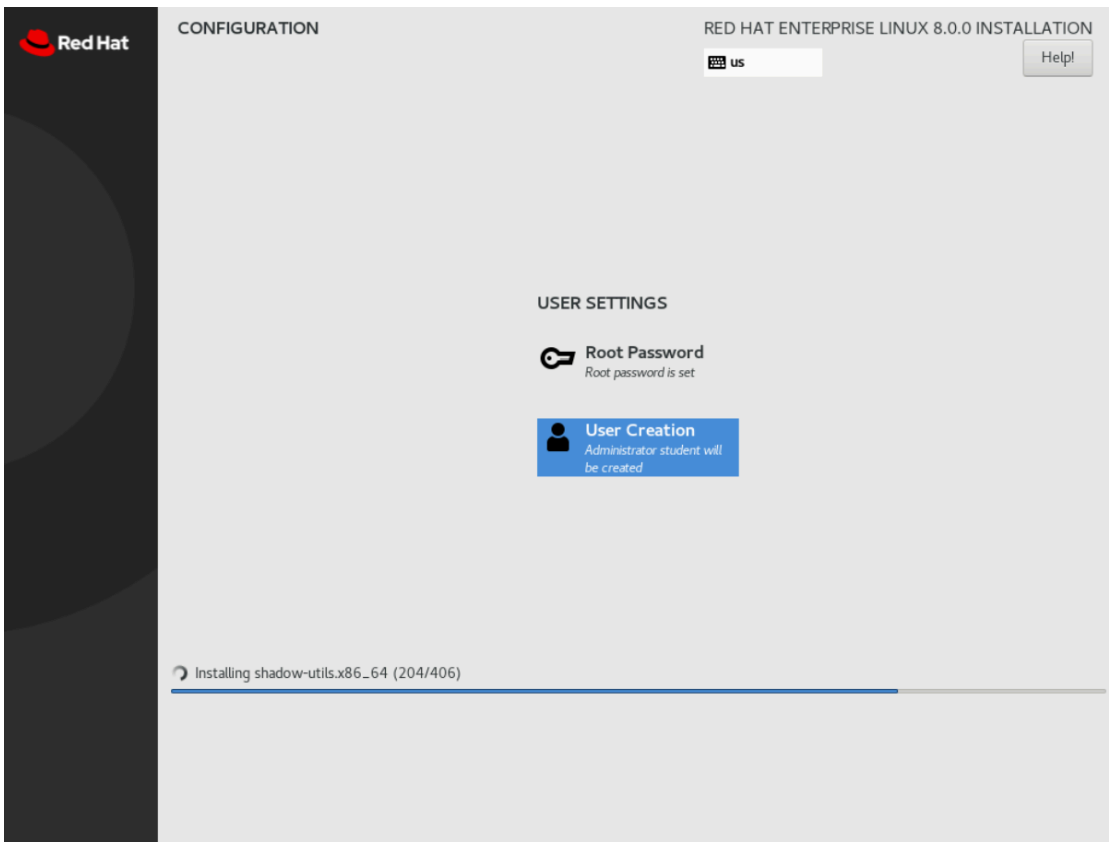

Figure 12.2: Setting the root password and creating a user

Click ***Reboot*** when the installation is done. Anaconda displays the ***Initial Setup*** screen, if a graphical desktop was installed. Accept the license information and optionally register the system with the subscription manager. You may skip system registration, and perform it later.

Troubleshooting the Installation

During a Red Hat Enterprise Linux 8 installation, Anaconda provides two virtual consoles. The first has five windows provided by the **`tmux`** software terminal multiplexer. You can access that console with **`Ctrl`**+**`Alt`**+**`F1`**. The second virtual console, which displays by default, shows the Anaconda graphical interface. You can access it with **`Ctrl`**+**`Alt`**+**`F6`**.

In the first virtual console, **`tmux`** provides a shell prompt in the second window. You may use it to enter commands to inspect and troubleshoot the system while the installation continues. The other windows provide diagnostic messages, logs and other information.

The following table lists the keystroke combinations to access the virtual consoles and the **`tmux`** windows. For **`tmux`**, the keyboard shortcuts are performed in two actions: press and release **`Ctrl`**+**`b`**, then press the number key of the window you want to access. With **`tmux`**, you can also use **`Alt`**+**`Tab`** to rotate the current focus between the windows.

KEY SEQUENCE	CONTENTS
`Ctrl+Alt+F1`	Access the **`tmux`** terminal multiplexer.
`Ctrl+b 1`	When in **`tmux`**, access the main information page for the installation process.
`Ctrl+b 2`	When in **`tmux`**, provide a root shell. Anaconda stores the installation log files in the **`/tmp`** file.
`Ctrl+b 3`	When in **`tmux`**, display the contents of the **`/tmp/anaconda.log`** file.
`Ctrl+b 4`	When in **`tmux`**, display the contents of the **`/tmp/storage.log`** file.
`Ctrl+b 5`	When in **`tmux`**, display the contents of the **`/tmp/program.log`** file.
`Ctrl+Alt+F6`	Access the Anaconda graphical interface.

NOTE

For compatibility with earlier Red Hat Enterprise Linux versions, the virtual consoles from **`Ctrl`**+**`Alt`**+**`F2`** through **`Ctrl`**+**`Alt`**+**`F5`** also present root shells during installation.

REFERENCES

For more information, refer to the *Installing and deploying RHEL* guide at https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/performing_a_standard_rhel_installation/index

GUIDED EXERCISE

INSTALLING RED HAT ENTERPRISE LINUX

In this exercise, you will reinstall one of your servers with a minimal installation of Red Hat Enterprise Linux.

OUTCOMES

You should be able to manually install Red Hat Enterprise Linux 8.

BEFORE YOU BEGIN

Log in as the `student` user on `workstation` using `student` as the password.

On `workstation`, run the **`lab installing-install start`** command. This command runs a start script to determine whether the `servera` machine is reachable on the network. It also adds a new entry in the GRUB2 menu to boot `servera` into the installation media.

```
[student@workstation ~]$ lab installing-install start
```

1. Access the `servera` console and reboot the system into the installation media.

   1.1. Locate the icon for the `servera` console, as appropriate for your classroom environment. Open the console.

   1.2. To reboot, send a **`Ctrl`**+**`Alt`**+**`Del`** to your system using the relevant keyboard, virtual, or menu entry.

   1.3. When the boot loader menu appears, select ***Install Red Hat Enterprise Linux 8***.

   1.4. Wait for the language selection window.

2. Keep the language selected by default and click ***Continue***.

3. Use automatic partitioning on the **`/dev/vda`** disk.

   3.1. Click ***Installation Destination***.

   3.2. Click on the first disk, **`vda`**, to select it. Click ***Done*** to use the default option of automatic partitioning.

   3.3. In the ***Installation Options*** window, click ***Reclaim space***. Because the **`/dev/vda`** disk already has partitions and file systems from the previous installation, this selection allows you to wipe the disk for the new installation. In the ***Reclaim Disk Space*** window, click ***Delete all*** then ***Reclaim space***.

4. Set the server host name to `servera.lab.example.com` and verify the configuration of the network interface.

   4.1. Click ***Network & Host Name***.

   4.2. In the ***Host Name*** field, enter **`servera.lab.example.com`**, and then click ***Apply***.

   4.3. Click ***Configure*** and then click the ***IPv4 Settings*** tab.

   4.4. Confirm that the network parameters are correct. The IP address is `172.25.250.10`, the netmask is `24`, and the gateway and the name server are each set to `172.25.250.254`. Click ***Save***.

   4.5. Confirm that the network interface is enabled by setting ***ON/OFF*** to **`ON`**.

   4.6. Click ***Done***.

5. Set the ***Installation Source*** field to `http://content.example.com/rhel8.0/x86_64/dvd`.

   5.1. Click ***Installation Source***.

   5.2. In the ***http://*** field, type **`content.example.com/rhel8.0/x86_64/dvd`**

   5.3. Click ***Done***.

6. Select the software required to run a minimal installation.

   6.1. Click ***Software Selection***.

   6.2. Select ***Minimal Install*** from the ***Base Environment*** list.

   6.3. Click ***Done***.

7. Configure the system's purpose.

   7.1. Click ***System Purpose***.

   7.2. Select a role of ***Red Hat Enterprise Linux Server***.

   7.3. Select an SLA level of ***Self-Support***.

   7.4. Select a usage of ***Development/Test***.

   7.5. Click ***Done***.

8. Click ***Begin Installation***.

9. While the installation progresses, set the password for `root` to `redhat`.

   9.1. Click ***Root Password***.

   9.2. Enter **`redhat`** in the ***Root Password*** field.

   9.3. Enter **`redhat`** in the ***Confirm*** field.

   9.4. The password is weak so you need to click ***Done*** twice.

10. While the installation progresses, add the `student` user.

    10.1. Click ***User Creation***.

    10.2. Enter **`student`** in the ***Full Name*** field.

    10.3. Check ***Make this user administrator*** so `student` can use **sudo** to run commands as `root`.

    10.4. Enter **`student`** in the ***Password*** field.

    10.5. Enter **`student`** in the ***Confirm password*** field.

    10.6. The password is weak so you need to click ***Done*** twice.

11. When the installation is complete, click ***Reboot***.

12. When the system displays the login prompt, log in as `student` with a password of `student`.

Finish

Use the appropriate method for your classroom environment to reset your `servera` machine.

This concludes the guided exercise.

AUTOMATING INSTALLATION WITH KICKSTART

OBJECTIVES

After completing this section, you should be able to:

- Explain Kickstart concepts and architecture.
- Create a Kickstart file with the Kickstart Generator website.
- Modify an existing Kickstart file with a text editor and check its syntax with **`ksvalidator`**.
- Publish a Kickstart file to the installer.
- Perform a network Kickstart installation.

CREATING A KICKSTART PROFILE

You can automate the installation of Red Hat Enterprise Linux using a feature called *Kickstart*. Using Kickstart, you specify everything Anaconda needs to complete an installation, including disk partitioning, network interface configuration, package selection, and other parameters, in a Kickstart text file. By referencing the text file, Anaconda performs the installation without further user interaction.

> **NOTE**
> Kickstart in Red Hat Enterprise Linux is similar to the Jumpstart facility in Oracle Solaris, or to using an unattended Setup answer file for Microsoft Windows.

Kickstart files begin with a list of commands that define how to install the target machine. Lines starting with **#** characters are comments to be ignored by the installer. Additional sections begin with a *directive*, recognized by a **%** first character, and end on a line with the **`%end`** directive.

The **`%packages`** section specifies the software to be installed on the target system. Specify individual packages by name (without versions). Package groups, specified by name or ID, are recognized by beginning them with an **@** character. Environment groups (groups of package groups) are recognized by beginning them with the **@^** characters. Specify modules, streams, and profiles with *@module:stream/profile* syntax.

Groups have mandatory, default, and optional components. Normally, Kickstart installs mandatory and default components. To exclude a package or package group from installation, precede it with a **-** character. However, excluded packages or package groups may still install if they are mandatory dependencies of other requested packages.

A Kickstart configuration commonly uses two additional sections, **`%pre`** and **`%post`**, which contain shell scripting commands that further configure the system. The **`%pre`** script is executed before any disk partitioning is done. Typically, this section is used only if actions are required to recognize or initialize a device before disk partitioning. The **`%post`** script is executed after the installation is otherwise completed.

You must specify the primary Kickstart commands before the **`%pre`**, **`%post`**, and **`%packages`** sections, but otherwise, you can place these sections in any order in the file.

KICKSTART FILE COMMANDS

Installation Commands

Define the installation source and how to perform the installation. Each is followed by an example.

- **`url`**: Specifies the URL pointing to the installation media.

```
url --url="http://classroom.example.com/content/rhel8.0/x86_64/dvd/"
```

- **`repo`**: Specifies where to find additional packages for installation. This option must point to a valid **`yum`** repository.

```
repo --name="appstream" --baseurl=http://classroom.example.com/content/rhel8.0/
x86_64/dvd/AppStream/
```

- **`text`**: Forces a text mode install.
- **`vnc`**: Allows the graphical installation to be viewed remotely over VNC.

```
vnc --password=redhat
```

Partitioning Commands

Define the devices and partitioning scheme to be used.

- **`clearpart`**: Removes partitions from the system prior to creation of new partitions. By default, no partitions are removed.

```
clearpart --all --drives=sda,sdb --initlabel
```

- **`part`**: Specifies the size, format, and name of a partition.

```
part /home --fstype=ext4 --label=homes --size=4096 --maxsize=8192 --grow
```

- **`autopart`**: Automatically creates a root partition, a swap partition, and an appropriate boot partition for the architecture. On large enough drives, this also creates a `/home` partition.
- **`ignoredisk`**: Controls Anaconda's access to disks attached to the system.

```
ignoredisk --drives=sdc
```

- **`bootloader`**: Defines where to install the bootloader.

```
bootloader --location=mbr --boot-drive=sda
```

- **`volgroup`**, **`logvol`**: Creates LVM volume groups and logical volumes.

```
part pv.01 --size=8192
volgroup myvg pv.01
logvol / --vgname=myvg --fstype=xfs --size=2048 --name=rootvol --grow
logvol /var --vgname=myvg --fstype=xfs --size=4096 --name=varvol
```

- **zerombr**: Initialize disks whose formatting is unrecognized.

Network Commands

Define the network features used by the host.

- **network**: Configures network information for target system. Activates network devices in installer environment.

```
network --device=eth0 --bootproto=dhcp
```

- **firewall**: Defines the firewall configuration for the target system.

```
firewall --enabled --service=ssh,http
```

Location and Security Commands

Configure settings related to security, language, and regions.

- **lang**: Sets the language to use during installation and the default language of the installed system. Required.

```
lang en_US.UTF-8
```

- **keyboard**: Sets the system keyboard type. Required.

```
keyboard --vckeymap=us --xlayouts=''
```

- **timezone**: Defines the timezone, NTP servers, and whether the hardware clock uses UTC.

```
timezone --utc --ntpservers=time.example.com Europe/Amsterdam
```

- **authselect**: Sets up authentication options. Options recognized by **authselect** are valid for this command. See authselect(8).
- **rootpw**: Defines the initial **root** password.

```
rootpw --plaintext redhat
or
rootpw --iscrypted $6$KUnFfrTzO8jv.PiH$YlBbOtXBkWzoMuRfb0.SpbQ....XDR1UuchoMG1
```

- **selinux**: Sets the SELinux mode for the installed system.

```
selinux --enforcing
```

- **services**: Modifies the default set of services to run under the default **systemd** target.

```
services --disabled=network,iptables,ip6tables --
enabled=NetworkManager,firewalld
```

- **group**, **user**: Create a local group or user on the system.

```
group --name=admins --gid=10001
user --name=jdoe --gecos="John Doe" --groups=admins --password=changeme --
plaintext
```

Miscellaneous Commands

Configure miscellaneous items related to logging during the installation and the host power state at completion.

- **`logging`**: This command defines how Anaconda will perform logging during the installation.

```
logging --host=loghost.example.com --level=info
```

- **`firstboot`**: If enabled, the Setup Agent starts the first time the system boots. The *initial-setup* package must be installed.

```
firstboot --disabled
```

- **`reboot`**, **`poweroff`**, **`halt`**: Specify the final action to take when the installation completes.

> **NOTE**
> The **`ksverdiff`** utility from the *pykickstart* package is useful for identifying changes in Kickstart file syntax between two versions of Red Hat Enterprise Linux or Fedora.
>
> For example, **`ksverdiff -f RHEL7 -t RHEL8`** identifies changes in syntax from RHEL 7 to RHEL 8. Available versions are listed in the top of the file **`/usr/lib/python3.6/site-packages/pykickstart/version.py`**.

EXAMPLE KICKSTART FILE

The first part of the file consists of the installation commands, such as disk partitioning and installation source.

```
#version=RHEL8
ignoredisk --only-use=vda
# System bootloader configuration
bootloader --append="console=ttyS0 console=ttyS0,115200n8 no_timer_check
 net.ifnames=0  crashkernel=auto" --location=mbr --timeout=1 --boot-drive=vda
# Clear the Master Boot Record
zerombr
# Partition clearing information
clearpart --all --initlabel
# Use text mode install
text
repo --name="appstream" --baseurl=http://classroom.example.com/content/rhel8.0/
x86_64/dvd/AppStream/
# Use network installation
url --url="http://classroom.example.com/content/rhel8.0/x86_64/dvd/"
# Keyboard layouts
# old format: keyboard us
```

```
# new format:
keyboard --vckeymap=us --xlayouts=''
# System language
lang en_US.UTF-8
# Root password
rootpw --plaintext redhat
# System authorization information
auth --enableshadow --passalgo=sha512
# SELinux configuration
selinux --enforcing
firstboot --disable
# Do not configure the X Window System
skipx
# System services
services --disabled="kdump,rhsmcertd" --enabled="sshd,rngd,chronyd"
# System timezone
timezone America/New_York --isUtc
# Disk partitioning information
part / --fstype="xfs" --ondisk=vda --size=10000
```

The second part contains the **%packages** section, detailing which packages and package groups should be installed, and which packages should not be installed.

```
%packages
@core
chrony
cloud-init
dracut-config-generic
dracut-norescue
firewalld
grub2
kernel
rsync
tar
-plymouth

%end
```

The last part contains any **%pre** and **%post** installation scripts.

```
%post --erroronfail

# For cloud images, 'eth0' _is_ the predictable device name, since
# we don't want to be tied to specific virtual (!) hardware
rm -f /etc/udev/rules.d/70*
ln -s /dev/null /etc/udev/rules.d/80-net-name-slot.rules

# simple eth0 config, again not hard-coded to the build hardware
cat > /etc/sysconfig/network-scripts/ifcfg-eth0 << EOF
DEVICE="eth0"
BOOTPROTO="dhcp"
ONBOOT="yes"
TYPE="Ethernet"
```

```
USERCTL="yes"
PEERDNS="yes"
IPV6INIT="no"
EOF

%end
```

NOTE
In a Kickstart file, missing required values cause the installer to interactively prompt for an answer or to abort the installation entirely.

KICKSTART INSTALLATION STEPS

To successfully automate installation of Red Hat Enterprise Linux, follow these steps:

1. Create a Kickstart file.
2. Publish the Kickstart file to the installer.
3. Boot Anaconda and point it to the Kickstart file.

CREATING A KICKSTART FILE

Use either of these methods to create a Kickstart file:

- Use the Kickstart Generator website.
- Use a text editor.

The Kickstart Generator website at `https://access.redhat.com/labs/kickstartconfig/` presents dialog boxes for user inputs, and creates a Kickstart directives text file with the user's choices. Each dialog box corresponds to the configurable items in the Anaconda installer.

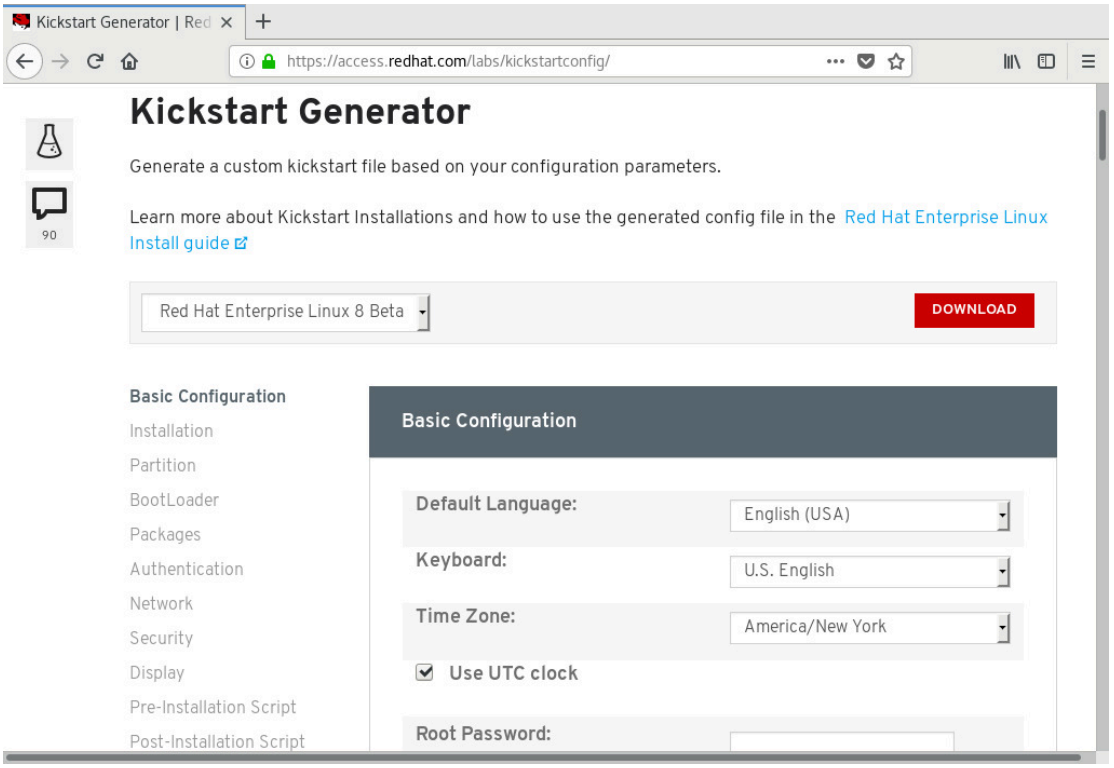

Figure 12.3: Basic Configuration with Kickstart Generator

NOTE
At the time of writing, the Kickstart Generator website did not provide Red Hat Enterprise Linux 8 as a menu option. Red Hat Enterprise Linux 8 Beta was a valid selection.

Creating a Kickstart file from scratch is typically too complex, but editing an existing Kickstart file is common and useful. Every installation creates a **`/root/anaconda-ks.cfg`** file containing the Kickstart directives used in the installation. This file makes a good starting point when creating Kickstart file manually.

`ksvalidator` is a utility that checks for syntax errors in a Kickstart file. It ensures that keywords and options are properly used, but it does not validate that URL paths, individual packages, or groups, nor any part of **`%post`** or **`%pre`** scripts will succeed. For instance, if the **`firewall --disabled`** directive is misspelled, **`ksvalidator`** could produce one of the following errors:

```
[user@host ~]$ ksvalidator /tmp/anaconda-ks.cfg
The following problem occurred on line 10 of the kickstart file:

Unknown command: frewall

[user@host ~]$ ksvalidator /tmp/anaconda-ks.cfg
The following problem occurred on line 10 of the kickstart file:

no such option: --dsabled
```

The *pykickstart* package provides **`ksvalidator`**.

PUBLISH THE KICKSTART FILE TO ANACONDA

Make the Kickstart file available to the installer by placing it in one of these locations:

- A network server available at install time using FTP, HTTP, or NFS.
- An available USB disk or CD-ROM.
- A local hard disk on the system to be installed.

The installer must access the Kickstart file to begin an automated installation. The most common automation method uses a network server such as an FTP, web, or NFS server. Network servers facilitate Kickstart file maintenance because changes can be made once, and then immediately used for multiple future installations.

Providing Kickstart files on USB or CD-ROM is also convenient. The Kickstart file can be embedded on the boot media used to start the installation. However, when the Kickstart file is changed, you must generate new installation media.

Providing the Kickstart file on a local disk allows you to quickly rebuild a system.

BOOT ANACONDA AND POINT IT TO THE KICKSTART FILE

Once a Kickstart method is chosen, the installer is told where to locate the Kickstart file by passing the **`inst.ks=`*`LOCATION`*** parameter to the installation kernel. Some examples:

- inst.ks=http://*server*/*dir*/*file*
- inst.ks=ftp://*server*/*dir*/*file*
- inst.ks=nfs:*server*:/*dir*/*file*
- inst.ks=hd:*device*:/*dir*/*file*
- inst.ks=cdrom:*device*

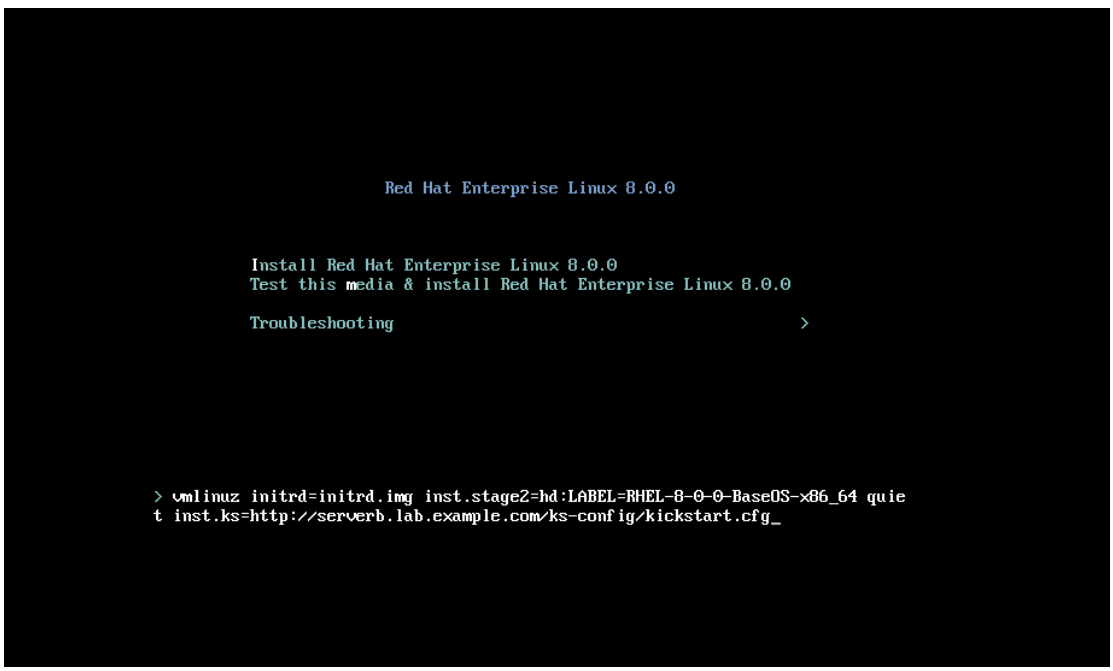

Figure 12.4: Specifying the Kickstart file location during installation

For virtual machine installations using the ***Virtual Machine Manager*** or **`virt-manager`**, the Kickstart URL can be specified in a box under ***URL Options***. When installing physical machines, boot using installation media and press the **`Tab`** key to interrupt the boot process. Add an **`inst.ks=`*`LOCATION`*** parameter to the installation kernel.

REFERENCES

Kickstart installation basics chapter in *Performing an advanced RHEL installation* at https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/performing_an_advanced_rhel_installation/kickstart-installation-basics_installing-rhel-as-an-experienced-user#kickstart-installation-basics_installing-rhel-as-an-experienced-user

Kickstart commands for installation program configuration and flow control section under *Appendix B. Kickstart commands and options reference* in *Performing an advanced RHEL installation* at https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/performing_an_advanced_rhel_installation/kickstart-installation-basics_installing-rhel-as-an-experienced-user#kickstart-commands-for-installation-program-configuration-and-flow-control_kickstart-commands-and-options-reference

Boot options chapter in *Performing an advanced RHEL installation* at https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/performing_an_advanced_rhel_installation/kickstart-installation-basics_installing-rhel-as-an-experienced-user#kickstart-and-advanced-boot-options_installing-rhel-as-an-experienced-user

GUIDED EXERCISE

AUTOMATING INSTALLATION WITH KICKSTART

In this lab, you will create a kickstart file and validate the syntax.

OUTCOMES

You should be able to:

- Create a kickstart file.
- Use **ksvalidator** to validate the kickstart file's syntax.

BEFORE YOU BEGIN

Log in as the `student` user on `workstation` using `student` as the password.

On `workstation`, run the **lab installing-kickstart start** command. This command runs a start script to determine whether the `servera` machine is reachable on the network. It also verifies that Apache is installed and configured on `servera`.

```
[student@workstation ~]$ lab installing-kickstart start
```

1. Use the **ssh** command to log in to `servera` as the `student` user. The systems are configured to use SSH keys for authentication, so a password is not required.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

2. Copy **/root/anaconda-ks.cfg** on `servera` to a file called **/home/student/kickstart.cfg** so that `student` can edit. Use the **sudo cat /root/anaconda-ks.cfg > ~/kickstart.cfg** command to copy the contents of **/root/anaconda-ks.cfg** to **/home/student/kickstart.cfg**. If **sudo** prompts for the password of the `student` user, use `student` as the password.

```
[student@servera ~]$ sudo cat /root/anaconda-ks.cfg > ~/kickstart.cfg
[sudo] password for student: student
```

3. Make the following changes to **/home/student/kickstart.cfg**.

   3.1. Comment out the reboot directive:

```
#reboot
```

3.2. Comment out the **repo** command for the BaseOS repository. Modify the **repo** command for AppStream to point to the classroom's AppStream repository:

```
#repo --name="koji-override-0" --baseurl=http://download-
node-02.eng.bos.redhat.com/rhel-8/devel/candidate-trees/RHEL-8/
RHEL-8.0.0-20190213.0/compose/BaseOS/x86_64/os
repo --name="appstream" --baseurl=http://classroom.example.com/
content/rhel8.0/x86_64/dvd/AppStream/
```

3.3. Change the **url** command to specify the classroom's HTTP installation source media:

```
url --url="http://classroom.example.com/content/rhel8.0/x86_64/dvd/"
```

3.4. Comment out the **network** command:

```
#network  --bootproto=dhcp --device=link --activate
```

3.5. Set the root password to `redhat`. Change the line that starts with **rootpw** to:

```
rootpw --plaintext redhat
```

3.6. Delete the line that uses the **auth** command and add the **authselect select sssd** line to set the `sssd` service as the identity and authentication source.

```
authselect select sssd
```

In Red Hat Enterprise Linux 8, the **authselect** command replaces the **authconfig** command.

3.7. Simplify the **services** command to look exactly like the following:

```
services --disabled="kdump,rhsmcertd" --enabled="sshd,rngd,chronyd"
```

3.8. Comment out the **part** and **reqpart** commands. Add the **autopart** command:

```
#reqpart
# Disk partitioning information
#part / --fstype="xfs" --ondisk=vda --size=8000
autopart
```

3.9. Delete all content between the **%post** section and its **%end**. Add the following line: **echo "Kickstarted on $(date)" >> /etc/issue**

The entire **%post** section should look like this.

```
%post --erroronfail
echo "Kickstarted on $(date)" >> /etc/issue
%end
```

3.10. Simplify the package specification to look exactly like the following:

```
%packages
@core
chrony
dracut-config-generic
dracut-norescue
firewalld
grub2
kernel
rsync
tar
httpd
-plymouth
%end
```

When you are finished editing the file save and exit.

4. Use the **ksvalidator** command to check the Kickstart file for syntax errors.

```
[student@servera ~]$ ksvalidator kickstart.cfg
```

5. Copy **kickstart.cfg** to the **/var/www/html/ks-config** directory.

```
[student@servera ~]$ sudo cp ~/kickstart.cfg /var/www/html/ks-config
```

6. Exit from servera.

```
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```

Finish

On workstation, run the **lab installing-kickstart finish** script to finish this exercise.

```
[student@workstation ~]$ lab installing-kickstart finish
```

This concludes the guided exercise.

INSTALLING AND CONFIGURING VIRTUAL MACHINES

OBJECTIVES

After completing this section, you should be able to install a virtual machine on your Red Hat Enterprise Linux server using Cockpit.

INTRODUCING KVM VIRTUALIZATION

Virtualization is a feature that allows a single physical machine to be divided into multiple *virtual machines (VM)*, each of which can run an independent operating system.

Red Hat Enterprise Linux 8 supports *KVM (Kernel-based Virtual Machine)*, a full virtualization solution built into the standard Linux kernel. KVM can run multiple Windows and Linux guest operating systems.

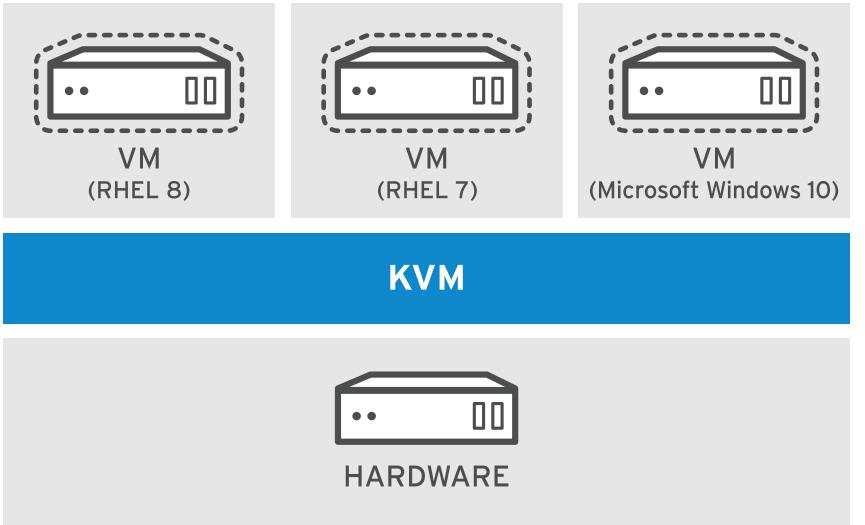

Figure 12.5: KVM virtualization

In Red Hat Enterprise Linux, manage KVM with the **`virsh`** command or with Cockpit's Virtual Machines tool.

KVM virtual machine technology is available across all Red Hat products, from standalone physical instances of Red Hat Enterprise Linux to the Red Hat OpenStack Platform:

- Physical hardware systems run Red Hat Enterprise Linux to provide KVM virtualization. Red Hat Enterprise Linux is typically a *thick* host, a system that supports VMs while also providing other local and network services, applications, and management functions.
- *Red Hat Virtualization (RHV)* provides a centralized web interface that allows administrators to manage an entire virtual infrastructure. It includes advanced features such as KVM migration, redundancy, and high availability. A *Red Hat Virtualization Hypervisor* is a tuned version of Red Hat Enterprise Linux dedicated to the singular purpose of provisioning and supporting VMs.
- *Red Hat OpenStack Platform (RHOSP)* provides the foundation to create, deploy, and scale a public or a private cloud.

Red Hat supports virtual machines running these operating systems:

- Red Hat Enterprise Linux 6 and later
- Microsoft Windows 10 and later
- Microsoft Windows Server 2016 and later

CONFIGURING A RED HAT ENTERPRISE LINUX PHYSICAL SYSTEM AS A VIRTUALIZATION HOST

Administrators can configure a Red Hat Enterprise Linux system as a virtualization host, appropriate for development, testing, training, or when needing to work in multiple operating systems at the same time.

Installing the Virtualization Tools

Install the *virt* Yum module to prepare a system to become a virtualization host.

```
[root@host ~]# yum module list virt
Name            Stream              Profiles            Summary
virt            rhel [d][e]         common [d]          Virtualization module

Hint: [d]efault, [e]nabled, [x]disabled, [i]nstalled
[root@host ~]# yum module install virt
...output omitted...
```

Verifying the System Requirements

KVM requires either an Intel processor with the Intel VT-x and Intel 64 extensions for x86-based systems or an AMD processor with the AMD-V and the AMD64 extensions. To verify your hardware and check the system requirements, use the **`virt-host-validate`** command.

```
[root@host ~]# virt-host-validate
  QEMU: Checking for hardware virtualization                 : PASS
  QEMU: Checking if device /dev/kvm exists                   : PASS
  QEMU: Checking if device /dev/kvm is accessible            : PASS
  QEMU: Checking if device /dev/vhost-net exists             : PASS
  QEMU: Checking if device /dev/net/tun exists               : PASS
  QEMU: Checking for cgroup 'memory' controller support      : PASS
  QEMU: Checking for cgroup 'memory' controller mount-point  : PASS
  QEMU: Checking for cgroup 'cpu' controller support         : PASS
  QEMU: Checking for cgroup 'cpu' controller mount-point     : PASS
  QEMU: Checking for cgroup 'cpuacct' controller support     : PASS
  QEMU: Checking for cgroup 'cpuacct' controller mount-point : PASS
  QEMU: Checking for cgroup 'cpuset' controller support      : PASS
  QEMU: Checking for cgroup 'cpuset' controller mount-point  : PASS
  QEMU: Checking for cgroup 'devices' controller support     : PASS
  QEMU: Checking for cgroup 'devices' controller mount-point : PASS
  QEMU: Checking for cgroup 'blkio' controller support       : PASS
  QEMU: Checking for cgroup 'blkio' controller mount-point   : PASS
  QEMU: Checking for device assignment IOMMU support         : PASS
```

The system must pass all of the validation items in order to be able to be a KVM host.

MANAGING VIRTUAL MACHINES WITH COCKPIT

The *virt* Yum module provides the **virsh** command to manage your virtual machines. The Cockpit tool provides a web console interface for KVM management and virtual machine creation.

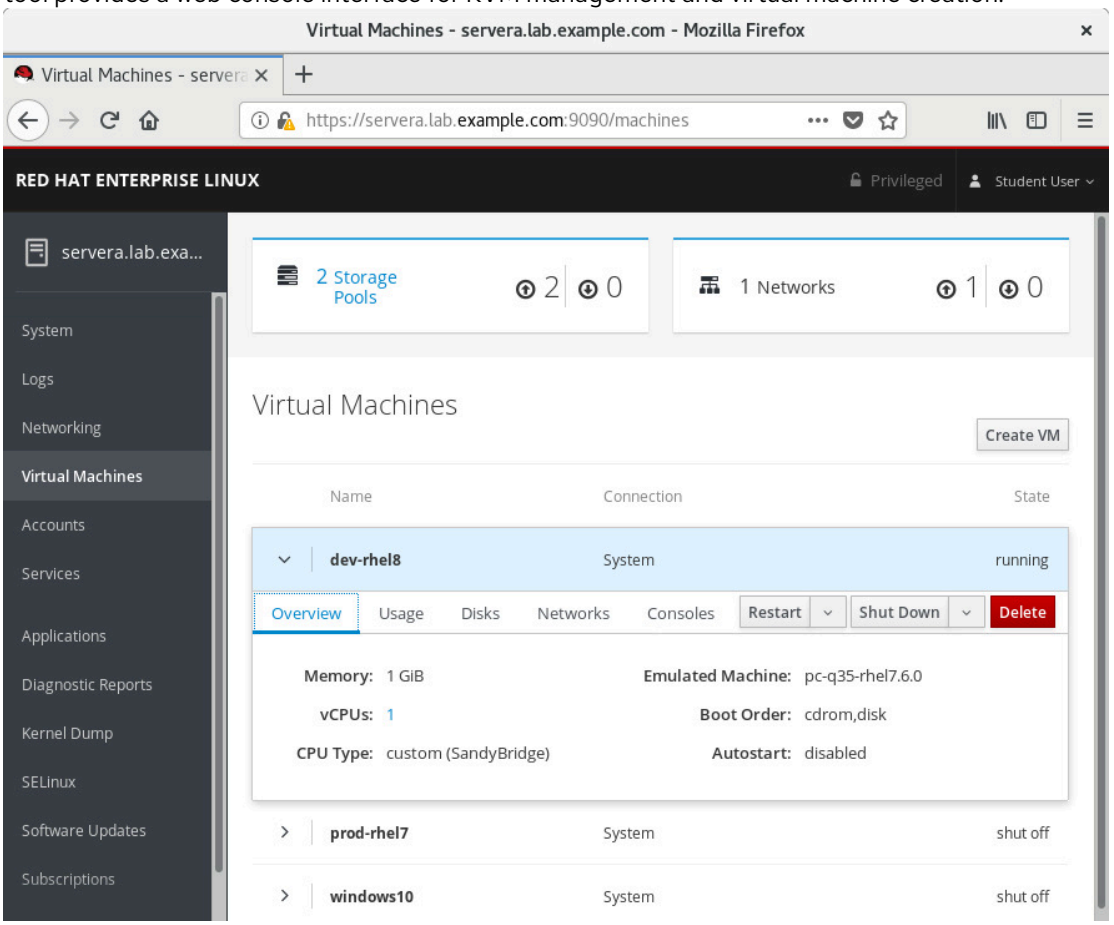

Figure 12.6: Managing virtual machines in Cockpit

Install the *cockpit-machines* package to add the ***Virtual Machines*** menu to Cockpit.

```
[root@host ~]# yum install cockpit-machines
```

If Cockpit is not already running, start and enable it.

```
[root@host ~]# systemctl enable --now cockpit.socket
```

To create a new virtual machine with Cockpit, access the ***Virtual Machines*** menu in the Cockpit web interface. From there, click ***Create VM*** and enter the VM configuration in the ***Create New Virtual Machine*** window.

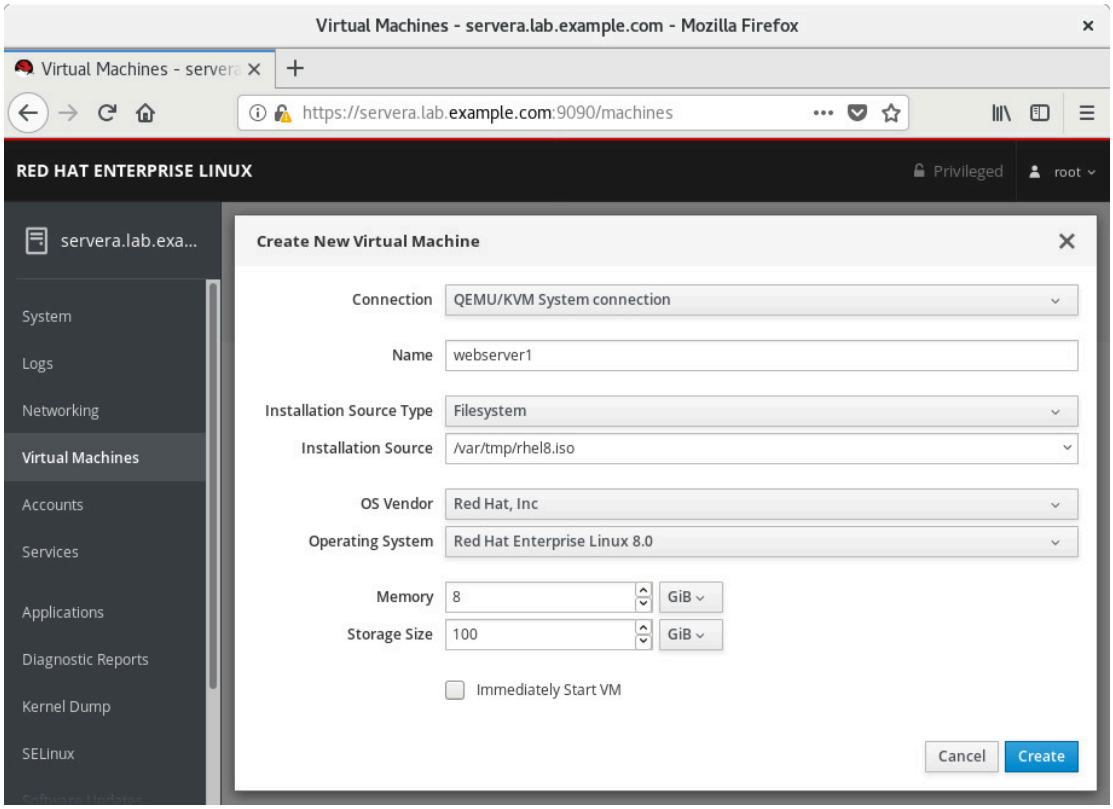

Figure 12.7: Creating a virtual machine in Cockpit

- ***Name*** sets a *domain* name for the virtual machine configuration. This name is unrelated to the network host name you give the system in the installed VM.
- ***Installation Source Type*** is the method for retrieving the installation ISO file. Choices include the local file system, or an HTTPS, FTP, or NFS URL.
- ***Installation Source*** provides the path to the installation source.
- ***OS Vendor*** and ***Operating System*** indicates the virtual machine's operating system. The virtualization layer presents hardware emulation compatible with the operating system chosen.
- ***Memory*** is the amount of RAM to make available to the new VM.
- ***Storage Size*** is the disk size to give the new VM. Associate additional disks with the VM after installation.
- ***Immediately Start VM*** indicates whether the VM should immediately start after you click ***Create***.

Click ***Create*** to create the VM and ***Install*** to start the operating system installation. Cockpit displays the VM console from which you can install the system.

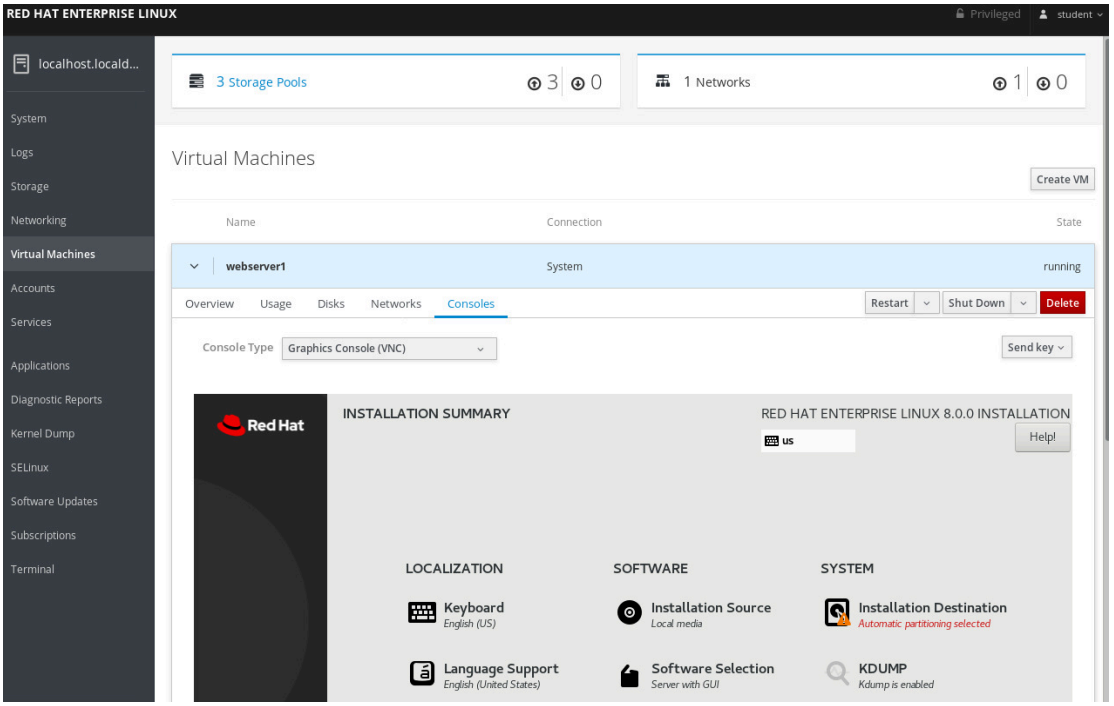

Figure 12.8: Installing the virtual machine OS

REFERENCES

For more information, refer to the *Configuring and managing virtualization* guide at https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/configuring_and_managing_virtualization/index

What is virtualization?
https://www.redhat.com/en/topics/virtualization/what-is-virtualization

QUIZ

INSTALLING AND CONFIGURING VIRTUAL MACHINES

Choose the correct answers to the following questions:

1. Which three guest operating systems does Red Hat support as KVM virtual machines? (Choose three.)

a. Fedora 28 and later
b. Red Hat Enterprise Linux 6 and later
c. CoreOS Container Linux 2023 and later
d. Microsoft Windows 7 SP1
e. Microsoft Windows 10 and later
f. Microsoft Windows Server 2016 and later

2. Which two components are required to configure your system as a virtualization host and manage virtual machines with Web Console? (Choose two.)

a. The *virt* Yum module
b. The *openstack* package group
c. The *cockpit-machines* package
d. The *Virtualization Platform* package group
e. The *kvm* yum module
f. The *cockpit-virtualization* package

3. Which command verifies that your system supports virtualization?

a. grep kvm /proc/cpuinfo
b. virsh validate
c. virt-host-validate
d. rhv-validate
e. cockpit-validate

4. Which two tools can you use to start and stop your virtual machines on a Red Hat Enterprise Linux system? (Choose two.)

a. vmctl
b. libvirtd
c. virsh
d. openstack
e. Web Console

SOLUTION

INSTALLING AND CONFIGURING VIRTUAL MACHINES

Choose the correct answers to the following questions:

1. Which three guest operating systems does Red Hat support as KVM virtual machines? (Choose three.)

a. Fedora 28 and later
b. Red Hat Enterprise Linux 6 and later
c. CoreOS Container Linux 2023 and later
d. Microsoft Windows 7 SP1
e. Microsoft Windows 10 and later
f. Microsoft Windows Server 2016 and later

2. Which two components are required to configure your system as a virtualization host and manage virtual machines with Web Console? (Choose two.)

a. The *virt* Yum module
b. The *openstack* package group
c. The *cockpit-machines* package
d. The *Virtualization Platform* package group
e. The *kvm* yum module
f. The *cockpit-virtualization* package

3. Which command verifies that your system supports virtualization?

a. grep kvm /proc/cpuinfo
b. virsh validate
c. virt-host-validate
d. rhv-validate
e. cockpit-validate

4. Which two tools can you use to start and stop your virtual machines on a Red Hat Enterprise Linux system? (Choose two.)

a. vmctl
b. libvirtd
c. virsh
d. openstack
e. Web Console

LAB

INSTALLING RED HAT ENTERPRISE LINUX

PERFORMANCE CHECKLIST

In this lab, you will create a kickstart file and perform a kickstart installation on `serverb`.

OUTCOMES

You should be able to:

- Create a kickstart file.
- Make the kickstart file available to the installer.
- Perform a kickstart installation.

BEFORE YOU BEGIN

Log in to `workstation` as `student` using `student` as the password.

On `workstation`, run the **`lab installing-review start`** command. This command runs a start script to determine whether the `servera` and `serverb` machines are reachable on the network, and configures Apache on `serverb`. It also configures the boot menu on `serverb` for the exercise to perform a kickstart installation.

```
[student@workstation ~]$ lab installing-review start
```

Prepare a kickstart file on `serverb` as specified and make it available at `http://serverb.lab.example.com/ks-config/kickstart.cfg`. Perform a kickstart installation on `servera` using the kickstart file you prepared.

1. On `serverb`, copy **`/root/anaconda-ks.cfg`** to **`/home/student/kickstart.cfg`**, so the `student` user can edit it.
2. Make the following changes to **`/home/student/kickstart.cfg`**.
   - Comment out the **`reboot`** command.
   - Comment out the **`repo`** command for the BaseOS repository. Modify the **`repo`** command for the AppStream repository to point to `http://classroom.example.com/content/rhel8.0/x86_64/dvd/AppStream/`. The repository name should be set to appstream.
   - Change the **`url`** command to use `http://classroom.example.com/content/rhel8.0/x86_64/dvd/` as the installation source.
   - Comment out the **`network`** command.
   - Change the **`rootpw`** command to use **`plaintext`** and set the root password to `redhat`.

- Delete the line that uses the **auth** command and add the **authselect select sssd** line to set the `sssd` service as the identity and authentication source.
- Simplify the **services** command so that only the `kdump` and `rhsmcertd` services are disabled. Leave only the `sshd`, `rngd`, and `chronyd` enabled.
- Add the **autopart** command. The **part** and **reqpart** commands already should be commented out.
- Simplify the %post section so that it only runs a script to append the text **Kickstarted on *DATE*** to the end of the **/etc/issue** file. *DATE* is variable information and should be generated by the script using the **date** command with no additional options.
- Simplify the **%package** section as follows: include the *@core*, *chrony*, *dracut-config-generic*, *dracut-norescue*, *firewalld*, *grub2*, *kernel*, *rsync*, *tar*, and *httpd* packages. Ensure that the *plymouth* package is not installed.

3. Validate the syntax of **kickstart.cfg**.
4. Make the **/home/student/kickstart.cfg** file available at `http://serverb.lab.example.com/ks-config/kickstart.cfg`
5. Return to the `workstation` system to check your work.

Evaluation

On `workstation`, run the **lab installing-review grade** script to grade this exercise. Reboot `servera` to perform a kickstart installation.

```
[student@workstation ~]$ lab installing-review grade
```

Correct any failures in **kickstart.cfg** being shared from the `serverb` web server by either modifying **/var/www/html/ks-config/kickstart.cfg** directly or by modifying **~/kickstart.cfg** and copying it to **/var/www/html/ks-config/**.

Reboot `servera` to perform a kickstart installation. At the GRUB menu, select **Kickstart Red Hat Enterprise Linux 8** and press **Enter**.

Finish

On `workstation`, run the **lab installing-review finish** script to finish this exercise. This script removes the web server configured on `serverb` during the exercise.

```
[student@workstation ~]$ lab installing-review finish
```

Reset the `servera` system to return it to the default state.

This concludes the lab.

SOLUTION

INSTALLING RED HAT ENTERPRISE LINUX

PERFORMANCE CHECKLIST

In this lab, you will create a kickstart file and perform a kickstart installation on `serverb`.

OUTCOMES

You should be able to:

- Create a kickstart file.
- Make the kickstart file available to the installer.
- Perform a kickstart installation.

BEFORE YOU BEGIN

Log in to `workstation` as `student` using `student` as the password.

On `workstation`, run the **`lab installing-review start`** command. This command runs a start script to determine whether the `servera` and `serverb` machines are reachable on the network, and configures Apache on `serverb`. It also configures the boot menu on `serverb` for the exercise to perform a kickstart installation.

```
[student@workstation ~]$ lab installing-review start
```

Prepare a kickstart file on `serverb` as specified and make it available at `http://serverb.lab.example.com/ks-config/kickstart.cfg`. Perform a kickstart installation on `servera` using the kickstart file you prepared.

1. On `serverb`, copy **`/root/anaconda-ks.cfg`** to **`/home/student/kickstart.cfg`**, so the `student` user can edit it.

   1.1. Use the **`ssh`** command to log in to `serverb` as the `student` user.

   ```
   [student@workstation ~]$ ssh student@serverb
   ...output omitted...
   [student@serverb ~]$
   ```

   1.2. Copy **`/root/anaconda-ks.cfg`** on `serverb` to a file called **`/home/student/kickstart.cfg`** so that `student` can edit. Use the **`sudo cat /root/anaconda-ks.cfg > ~/kickstart.cfg`** command to copy the contents of **`/root/anaconda-ks.cfg`** to **`/home/student/kickstart.cfg`**. If **`sudo`** prompts for the password of the `student` user, use `student` as the password.

```
[student@serverb ~]$ sudo cat /root/anaconda-ks.cfg > ~/kickstart.cfg
[sudo] password for student: student
```

2. Make the following changes to **/home/student/kickstart.cfg**.

   - Comment out the **reboot** command.
   - Comment out the **repo** command for the BaseOS repository. Modify the **repo** command for the AppStream repository to point to `http://classroom.example.com/`

`content/rhel8.0/x86_64/dvd/AppStream/`. The repository name should be set to appstream.

- Change the **`url`** command to use `http://classroom.example.com/content/rhel8.0/x86_64/dvd/` as the installation source.
- Comment out the **`network`** command.
- Change the **`rootpw`** command to use **`plaintext`** and set the root password to `redhat`.
- Delete the line that uses the **`auth`** command and add the **`authselect select sssd`** line to set the `sssd` service as the identity and authentication source.
- Simplify the **`services`** command so that only the `kdump` and `rhsmcertd` services are disabled. Leave only the `sshd`, `rngd`, and `chronyd` enabled.
- Add the **`autopart`** command. The **`part`** and **`reqpart`** commands already should be commented out.
- Simplify the %post section so that it only runs a script to append the text **`Kickstarted on DATE`** to the end of the **`/etc/issue`** file. *DATE* is variable information and should be generated by the script using the **`date`** command with no additional options.
- Simplify the **`%package`** section as follows: include the *@core*, *chrony*, *dracut-config-generic*, *dracut-norescue*, *firewalld*, *grub2*, *kernel*, *rsync*, *tar*, and *httpd* packages. Ensure that the *plymouth* package is not installed.

2.1. Comment out the reboot directive:

```
#reboot
```

2.2. The **`repo`** command is found twice in **`kickstart.cfg`**. Comment out the **`repo`** command for the BaseOS repository. Modify the **`repo`** command for the AppStream repository to point to the classroom's AppStream repository:

```
#repo --name="koji-override-0" --baseurl=http://download-
node-02.eng.bos.redhat.com/rhel-8/devel/candidate-trees/RHEL-8/
RHEL-8.0.0-20190213.0/compose/BaseOS/x86_64/os
repo --name="appstream" --baseurl=http://classroom.example.com/content/
rhel8.0/x86_64/dvd/AppStream/
```

2.3. Change the **`url`** command to specify the HTTP installation source media used in the classroom:

```
url --url="http://classroom.example.com/content/rhel8.0/x86_64/dvd/"
```

2.4. Comment out the **`network`** command:

```
#network  --bootproto=dhcp --device=link --activate
```

2.5. Set the root password to `redhat`. Change the line that starts with **`rootpw`** to:

```
rootpw --plaintext redhat
```

2.6. Delete the line that uses the **auth** command and add the **authselect select sssd** line to set the `sssd` service as the identity and authentication source.

```
authselect select sssd
```

2.7. Simplify the **services** command to look exactly like the following:

```
services --disabled="kdump,rhsmcertd" --enabled="sshd,rngd,chronyd"
```

2.8. Comment out the **part** and **reqpart** commands. Add the **autopart** command:

```
#reqpart
# Disk partitioning information
#part / --fstype="xfs" --ondisk=vda --size=8000
autopart
```

2.9. Delete all content between the **%post** section and its **%end**. Add the following line: **echo "Kickstarted on $(date)" >> /etc/issue**

The entire **%post** section should look like this.

```
%post --erroronfail
echo "Kickstarted on $(date)" >> /etc/issue
%end
```

2.10. Simplify the package specification to look exactly like the following:

```
%packages
@core
chrony
dracut-config-generic
dracut-norescue
firewalld
grub2
kernel
rsync
tar
httpd
-plymouth
%end
```

3. Validate the syntax of **kickstart.cfg**.

3.1. Use the **ksvalidator** command to check the Kickstart file for syntax errors.

```
[student@serverb ~]$ ksvalidator kickstart.cfg
```

4. Make the **/home/student/kickstart.cfg** file available at `http://serverb.lab.example.com/ks-config/kickstart.cfg`

   4.1. Copy **kickstart.cfg** to the **/var/www/html/ks-config/** directory.

```
[student@serverb ~]$ sudo cp ~/kickstart.cfg /var/www/html/ks-config
```

5. Return to the `workstation` system to check your work.

   5.1. Exit from `serverb`.

```
[student@serverb ~]$ exit
logout
Connection to serverb closed.
[student@workstation ~]$
```

Evaluation

On `workstation`, run the **lab installing-review grade** script to grade this exercise. Reboot `servera` to perform a kickstart installation.

```
[student@workstation ~]$ lab installing-review grade
```

Correct any failures in **kickstart.cfg** being shared from the `serverb` web server by either modifying **/var/www/html/ks-config/kickstart.cfg** directly or by modifying **~/kickstart.cfg** and copying it to **/var/www/html/ks-config/**.

Reboot `servera` to perform a kickstart installation. At the GRUB menu, select **Kickstart Red Hat Enterprise Linux 8** and press **Enter**.

Finish

On `workstation`, run the **lab installing-review finish** script to finish this exercise. This script removes the web server configured on `serverb` during the exercise.

```
[student@workstation ~]$ lab installing-review finish
```

Reset the `servera` system to return it to the default state.

This concludes the lab.

SUMMARY

In this chapter, you learned:

- The RHEL 8 binary DVD includes Anaconda and all repositories required for installation.
- The RHEL 8 boot ISO includes the Anaconda installer, accessing repositories over the network during installation.
- The Kickstart system performs unattended installations.
- Kickstart files can be created using the Kickstart Generator website or by copying and editing **`/root/anaconda-ks.cfg`**.
- The *virt* Yum module provides the packages for a RHEL system to become a virtualization host.
- The *cockpit-machines* package adds the ***Virtual Machines*** menu to Cockpit.

CHAPTER 13

COMPREHENSIVE REVIEW

GOAL

Review tasks from *Red Hat System Administration II*

OBJECTIVES

- Review tasks from *Red Hat System Administration II*

SECTIONS

- Comprehensive Review

LAB

- Lab: Fixing Boot Issues and Maintaining Servers
- Lab: Configuring and Managing File Systems and Storage
- Lab: Configuring and Managing Server Security

COMPREHENSIVE REVIEW

OBJECTIVES

After completing this section, you should have reviewed and refreshed the knowledge and skills learned in *Red Hat System Administration II*.

REVIEWING *RED HAT SYSTEM ADMINISTRATION II*

Before beginning the comprehensive review for this course, you should be comfortable with the topics covered in each chapter.

You can refer to earlier sections in the textbook for extra study.

Chapter 1, *Improving Command-line Productivity*

Run commands more efficiently by using advanced features of the Bash shell, shell scripts, and various utilities provided by Red Hat Enterprise Linux.

- Automate sequences of commands by writing a simple shell script.
- Efficiently run commands over lists of items in a script or from the command-line using for loops and conditionals.
- Find text matching a pattern in log files and command output using the **`grep`** command and regular expressions.

Chapter 2, *Scheduling Future Tasks*

Schedule tasks to automatically execute in the future.

- Set up a command that runs once at some point in the future.
- Schedule commands to run on a repeating schedule using a user's crontab file.
- Schedule commands to run on a repeating schedule using the system crontab file and directories.
- Enable and disable systemd timers, and configure a timer that manages temporary files.

Chapter 3, *Tuning System Performance*

Improve system performance by setting tuning parameters and adjusting scheduling priority of processes.

- Optimize system performance by selecting a tuning profile managed by the tuned daemon.
- Prioritize or de-prioritize specific processes with the nice and renice commands.

Chapter 4, *Controlling Access to Files with ACLs*

Interpret and set Access Control Lists (ACLs) on files to handle situations requiring complex user and group access permissions.

- Describe use cases for ACLs, identify files that have ACLs set, and interpret the effect of those ACLs.
- Set and remove ACLs on files and define default ACLs automatically set by a directory on newly created files.

Chapter 5, *Managing SELinux Security*

Protect and manage the security of a server by using SELinux.

- Describe how SELinux protects resources and how to select the enforcement mode.
- Configure a file's SELinux context to control how processes interact with that file.
- Configure SELinux booleans to allow runtime policy changes for varying access needs.
- Investigate SELinux log messages and troubleshoot SELinux AVC denials.

Chapter 6, *Managing Basic Storage*

Create and manage storage devices, partitions, file systems, and swap spaces from the command line.

- Create storage partitions, format them with file systems, and mount them for use.
- Create and manage swap spaces to supplement physical memory.

Chapter 7, *Managing Logical Volumes*

Create and manage logical volumes containing file systems and swap spaces from the command line.

- Create and manage logical volumes from storage devices, and format them with file systems or prepare them with swap spaces.
- Add and remove storage assigned to volume groups, and nondestructively extend the size of a logical volume formatted with a file system.

Chapter 8, *Implementing Advanced Storage Features*

Manage storage using the Stratis local storage management system and use the VDO volumes to optimize storage space in use.

- Manage multiple storage layers using Stratis local storage management.
- Optimize the use of storage space by using VDO to compress and deduplicate data on storage devices.

Chapter 9, *Accessing Network-Attached Storage*

Access network-attached storage using the NFS protocol.

- Mount, use, and unmount an NFS export from the command line and at boot time.
- Configure the automounter with direct and indirect maps to automatically mount an NFS file system on demand, and unmount it when it is no longer in use.
- Configure an NFS client to use NFSv4 using the new **`nfsconf`** tool.

Chapter 10, *Controlling the Boot Process*

Manage the boot process to control services offered and to troubleshoot and repair problems.

- Describe the Red Hat Enterprise Linux boot process, set the default target used when booting, and boot a system to a non-default target.
- Log in to a system and change the root password when the current root password has been lost.
- Manually repair file system configuration or corruption issues that stop the boot process.

Chapter 11, *Managing Network Security*

Control network connections to services using the system firewall and SELinux rules.

- Accept or reject network connections to system services using firewalld rules.
- Control whether network services can use specific networking ports by managing SELinux port labels.

Chapter 12, *Installing Red Hat Enterprise Linux*

Install Red Hat Enterprise Linux on servers and virtual machines.

- Install Red Hat Enterprise Linux on a server.
- Automate the installation process using Kickstart.
- Install a virtual machine on your Red Hat Enterprise Linux server using Cockpit.

LAB

FIXING BOOT ISSUES AND MAINTAINING SERVERS

In this review, you will troubleshoot and repair boot problems and update the system default target. You will also schedule tasks to run on a repeating schedule as a normal user.

OUTCOMES

You should be able to:

- Diagnose issues and recover the system from emergency mode.
- Change the default target from `graphical.target` to `multi-user.target`.
- Schedule recurring jobs to run as a normal user.

BEFORE YOU BEGIN

Copy any files or work you wish to keep to other systems before resetting. Reset the `workstation`, `servera`, and `serverb` systems now.

Log in to `workstation` as `student` using `student` as the password.

On `workstation`, run **`lab rhcsa-compreview1 start`** to start the comprehensive review. This script creates the necessary files to set up the environment correctly.

```
[student@workstation ~]$ lab rhcsa-compreview1 start
```

INSTRUCTIONS

Perform the following tasks on `serverb` to complete the comprehensive review:

- On `workstation`, run the **`lab rhcsa-compreview1 break1`** command. This break script causes the boot process to fail on `serverb`. It also sets a longer timeout on the `GRUB2` menu to help interrupt the boot process, and reboots `serverb`.

  Troubleshoot the possible cause and repair the boot failure. The fix must ensure that `serverb` reboots without intervention. Use `redhat` as the password of the superuser, when required.

- On `workstation`, run the **`lab rhcsa-compreview1 break2`** command. This break script causes the default target to switch from the `multi-user` target to the `graphical` target on `serverb`. It also sets a longer timeout for the `GRUB2` menu to help interrupt the boot process, and reboots `serverb`.

  On `serverb`, fix the default target to use the `multi-user` target. The default target settings must persist after reboot without manual intervention.

  Use the **`sudo`** command, as the `student` user with `student` as the password, for performing privileged commands.

- Schedule a recurring job as the `student` user that executes the **`/home/student/backup-home.sh`** script on an hourly basis between 7 p.m. and 9 p.m. on all days except Saturday and Sunday.

  Download the backup script from http://materials.example.com/labs/backup-home.sh. The **`backup-home.sh`** backup script backs up the **`/home/student`** directory from `serverb` to `servera` in the **`/home/student/serverb-backup`** directory. Use the **`backup-home.sh`** script to schedule the recurring job as the `student` user on `serverb`.

- Reboot the system and wait for the boot to complete before grading.

Evaluation

On `workstation`, run the **`lab rhcsa-compreview1 grade`** script to confirm success on this exercise. Correct any reported failures and rerun the script until successful.

```
[student@workstation ~]$ lab rhcsa-compreview1 grade
```

Finish

On `workstation`, run **`lab rhcsa-compreview1 finish`** to complete this exercise. This script deletes the files and resources created throughout the exercise and ensures that the environment is clean.

```
[student@workstation ~]$ lab rhcsa-compreview1 finish
```

Save any files or work you want to keep to other systems, and then reset `workstation`, `servera`, and `serverb` before the next exercise.

This concludes the comprehensive review.

SOLUTION

FIXING BOOT ISSUES AND MAINTAINING SERVERS

In this review, you will troubleshoot and repair boot problems and update the system default target. You will also schedule tasks to run on a repeating schedule as a normal user.

OUTCOMES

You should be able to:

- Diagnose issues and recover the system from emergency mode.
- Change the default target from `graphical.target` to `multi-user.target`.
- Schedule recurring jobs to run as a normal user.

BEFORE YOU BEGIN

Copy any files or work you wish to keep to other systems before resetting. Reset the `workstation`, `servera`, and `serverb` systems now.

Log in to `workstation` as `student` using `student` as the password.

On `workstation`, run **`lab rhcsa-compreview1 start`** to start the comprehensive review. This script creates the necessary files to set up the environment correctly.

```
[student@workstation ~]$ lab rhcsa-compreview1 start
```

INSTRUCTIONS

Perform the following tasks on `serverb` to complete the comprehensive review:

- On `workstation`, run the **`lab rhcsa-compreview1 break1`** command. This break script causes the boot process to fail on `serverb`. It also sets a longer timeout on the `GRUB2` menu to help interrupt the boot process, and reboots `serverb`.

  Troubleshoot the possible cause and repair the boot failure. The fix must ensure that `serverb` reboots without intervention. Use `redhat` as the password of the superuser, when required.

- On `workstation`, run the **`lab rhcsa-compreview1 break2`** command. This break script causes the default target to switch from the `multi-user` target to the `graphical` target on `serverb`. It also sets a longer timeout for the `GRUB2` menu to help interrupt the boot process, and reboots `serverb`.

  On `serverb`, fix the default target to use the `multi-user` target. The default target settings must persist after reboot without manual intervention.

  Use the **`sudo`** command, as the `student` user with `student` as the password, for performing privileged commands.

- Schedule a recurring job as the `student` user that executes the **`/home/student/backup-home.sh`** script on an hourly basis between 7 p.m. and 9 p.m. on all days except Saturday and Sunday.

  Download the backup script from http://materials.example.com/labs/backup-home.sh. The **`backup-home.sh`** backup script backs up the **`/home/student`** directory from `serverb` to `servera` in the **`/home/student/serverb-backup`** directory. Use the **`backup-home.sh`** script to schedule the recurring job as the `student` user on `serverb`.

- Reboot the system and wait for the boot to complete before grading.

1. On `workstation`, run the **`lab rhcsa-compreview1 break1`** command.

```
[student@workstation ~]$ lab rhcsa-compreview1 break1
```

2. After `serverb` boots up, access the console and notice that the boot process stopped early. Take a minute to speculate about a possible cause for this behavior.

   2.1. Locate the icon for the `serverb` console, as appropriate for your classroom environment. Open the console.

   2.2. Looking at the error, it appears that at least parts of the system are still functioning.

   2.3. Press **`Ctrl`**+**`Alt`**+**`Del`** to reboot `serverb`.

   When the boot-loader menu appears, press any key except **`Enter`** to interrupt the countdown.

   2.4. Edit the default boot-loader entry, in memory, to log in to emergency mode.

   Press **`e`** to edit the current entry.

   2.5. Use the cursor keys to navigate to the line that starts with **`linux`**. Append **`systemd.unit=emergency.target`** to the end of the line.

   2.6. Press **`Ctrl`**+**`x`** to boot using the modified configuration.

   2.7. Log in to emergency mode. The `root` password is `redhat`.

```
Give root password for maintenance
(or press Control-D to continue): redhat
[root@serverb ~]#
```

3. Remount the **`/`** file system read/write. Use the **`mount -a`** command to attempt to mount all the other file systems.

   3.1. Remount the **`/`** file system read/write to edit the file system.

```
[root@serverb ~]# mount -o remount,rw /
```

   3.2. Use the **`mount -a`** command to attempt to mount all the other file systems. Notice that one of the file systems cannot be mounted.

```
[root@serverb ~]# mount -a
mount: /FakeMount: can't find UUID=fake.
```

3.3. Edit **/etc/fstab** to fix the issue. Remove or comment out the incorrect line.

```
[root@serverb ~]# vim /etc/fstab
...output omitted...
#UUID=fake     /FakeMount  xfs   defaults    0 0
```

3.4. Update `systemd` for the system to register the new **/etc/fstab** configuration.

```
[root@serverb ~]# systemctl daemon-reload
[ 206.828912] systemd[1]: Reloading.
```

3.5. Verify that **/etc/fstab** is now correct by attempting to mount all entries.

```
[root@serverb ~]# mount -a
```

3.6. Reboot `serverb` and wait for the boot to complete. The system should now boot normally.

```
[root@serverb ~]# systemctl reboot
```

4. On `workstation`, run the **lab rhcsa-compreview1 break2** command.

```
[student@workstation ~]$ lab rhcsa-compreview1 break2
```

Wait for the reboot to complete before proceeding.

5. On `serverb`, switch to the `multi-user` target. Set the default target to `multi-user`. Use the **sudo** command to run any required administrative command and if prompted, use `student` as the password.

5.1. From `workstation`, open an SSH session to `serverb` as the `student` user.

```
[student@workstation ~]$ ssh student@serverb
...output omitted...
[student@serverb ~]$
```

5.2. As the `student` user on `serverb`, determine the default target.

```
[student@serverb ~]$ systemctl get-default
graphical.target
```

5.3. Switch to the `multi-user` target. Use the **sudo** command and if prompted, use `student` as the password.

```
[student@serverb ~]$ sudo systemctl isolate multi-user.target
[sudo] password for student: student
```

5.4. Set `serverb` to use the `multi-user` target as the default target.

```
[student@serverb ~]$ sudo systemctl set-default multi-user.target
Removed /etc/systemd/system/default.target.
Created symlink /etc/systemd/system/default.target -> /usr/lib/systemd/
system/multi-user.target.
```

5.5. Reboot `serverb` to verify that the `multi-user` target is set as the default target.

```
[student@serverb ~]$ sudo systemctl reboot
Connection to serverb closed by remote host.
Connection to serverb closed.
[student@workstation ~]$
```

5.6. After reboot, open an SSH session to `serverb` as the `student` user. Verify that the `multi-user` target is set as the default target.

```
[student@workstation ~]$ ssh student@serverb
...output omitted...
[student@serverb ~]$ systemctl get-default
multi-user.target
```

6. Schedule a recurring job as the `student` user that executes the **`/home/student/backup-home.sh`** script on an hourly basis between 7 p.m. and 9 p.m. on all days except Saturday and Sunday.

   Use the **`backup-home.sh`** script to schedule the recurring job. Download the backup script from http://materials.example.com/labs/backup-home.sh.

   6.1. On `serverb`, download the backup script from http://materials.example.com/labs/backup-home.sh. Use **`chmod`** to make the backup script executable.

```
[student@serverb ~]$ wget http://materials.example.com/labs/backup-
home.sh
...output omitted...
[student@serverb ~]$ chmod +x backup-home.sh
```

   6.2. Use the **`crontab -e`** command to open the crontab file using the default text editor.

```
[student@serverb ~]$ crontab -e
```

   6.3. Edit the file to add the following line:

```
0 19-21 * * Mon-Fri /home/student/backup-home.sh
```

   Save the changes and exit the editor.

   6.4. Use the **`crontab -l`** command to list the scheduled recurring jobs.

```
[student@serverb ~]$ crontab -l
0 19-21 * * Mon-Fri /home/student/backup-home.sh
```

7. Reboot `serverb` and wait for the boot to complete before grading.

```
[student@serverb ~]$ sudo systemctl reboot
[sudo] password for student: student
Connection to serverb closed by remote host.
Connection to serverb closed.
[student@workstation ~]$
```

Evaluation

On `workstation`, run the **`lab rhcsa-compreview1 grade`** script to confirm success on this exercise. Correct any reported failures and rerun the script until successful.

```
[student@workstation ~]$ lab rhcsa-compreview1 grade
```

Finish

On `workstation`, run **`lab rhcsa-compreview1 finish`** to complete this exercise. This script deletes the files and resources created throughout the exercise and ensures that the environment is clean.

```
[student@workstation ~]$ lab rhcsa-compreview1 finish
```

Save any files or work you want to keep to other systems, and then reset `workstation`, `servera`, and `serverb` before the next exercise.

This concludes the comprehensive review.

LAB

CONFIGURING AND MANAGING FILE SYSTEMS AND STORAGE

In this review, you will create an LVM logical volume, mount a network file system, create a swap partition that is automatically activated at boot, configure temporary unused files to be cleaned from the system, and use ACLs to protect a directory.

OUTCOMES

You should be able to:

- Create an LVM logical volume.
- Mount a network file system.
- Create a swap partition that is automatically activated at boot.
- Configure temporary unused files to be cleaned from the system.
- Use ACLs to protect a directory.

BEFORE YOU BEGIN

Copy any files or work you wish to keep to other systems before resetting. Reset the `workstation`, `servera`, and `serverb` systems now, unless you just finished resetting them at the end of the last exercise.

Log in to `workstation` as `student` using `student` as the password.

On `workstation`, run **`lab rhcsa-compreview2 start`** to start the comprehensive review. This script creates the necessary files to set up the environment correctly.

```
[student@workstation ~]$ lab rhcsa-compreview2 start
```

INSTRUCTIONS

Perform the following tasks on `serverb` to complete the comprehensive review.

- Configure a new 1 GiB logical volume called `vol_home` in a new 2 GiB volume group called `extra_storage`. Use the unpartitioned `/dev/vdb` disk to create partitions.
- The logical volume `vol_home` should be formatted with the XFS file-system type, and mounted persistently on **`/home-directories`**.
- Ensure that the network file system called **`/share`** is persistently mounted on **`/local-share`** across reboot. The NFS server `servera.lab.example.com` exports the **`/share`** network file system. The NFS export path is `servera.lab.example.com:/share`.
- Create a new 512 MiB partition on the `/dev/vdc` disk to be used as swap space. This swap space must be automatically activated at boot.
- Create a new group called `production`. Create the `production1`, `production2`, `production3`, and `production4` users. Ensure that they use the new group called `production` as their supplementary group.

- Configure your system so that it uses a new directory called **/run/volatile** to store temporary files. Files in this directory should be subject to time based cleanup if they are not accessed for more than 30 seconds. The octal permissions for the directory must be **0700**. Make sure that you use the **/etc/tmpfiles.d/volatile.conf** file to configure the time based cleanup for the files in **/run/volatile**.
- Create the new directory called **/webcontent**. Both the owner and group of the directory should be `root`. The group members of `production` should be able to read and write to this directory. The `production1` user should only be able to read this directory. These permissions should apply to all new files and directories created under the **/webcontent** directory.

Evaluation

On `workstation`, run the **lab rhcsa-compreview2 grade** script to confirm success on this exercise. Correct any reported failures and rerun the script until successful.

```
[student@workstation ~]$ lab rhcsa-compreview2 grade
```

Finish

On `workstation`, run **lab rhcsa-compreview2 finish** to complete this exercise. This script deletes the files and resources created throughout the exercise and ensures that the environment is clean.

```
[student@workstation ~]$ lab rhcsa-compreview2 finish
```

This concludes the comprehensive review.

SOLUTION

CONFIGURING AND MANAGING FILE SYSTEMS AND STORAGE

In this review, you will create an LVM logical volume, mount a network file system, create a swap partition that is automatically activated at boot, configure temporary unused files to be cleaned from the system, and use ACLs to protect a directory.

OUTCOMES

You should be able to:

- Create an LVM logical volume.
- Mount a network file system.
- Create a swap partition that is automatically activated at boot.
- Configure temporary unused files to be cleaned from the system.
- Use ACLs to protect a directory.

BEFORE YOU BEGIN

Copy any files or work you wish to keep to other systems before resetting. Reset the `workstation`, `servera`, and `serverb` systems now, unless you just finished resetting them at the end of the last exercise.

Log in to `workstation` as `student` using `student` as the password.

On `workstation`, run **`lab rhcsa-compreview2 start`** to start the comprehensive review. This script creates the necessary files to set up the environment correctly.

```
[student@workstation ~]$ lab rhcsa-compreview2 start
```

INSTRUCTIONS

Perform the following tasks on `serverb` to complete the comprehensive review.

- Configure a new 1 GiB logical volume called `vol_home` in a new 2 GiB volume group called `extra_storage`. Use the unpartitioned `/dev/vdb` disk to create partitions.
- The logical volume `vol_home` should be formatted with the XFS file-system type, and mounted persistently on **`/home-directories`**.
- Ensure that the network file system called **`/share`** is persistently mounted on **`/local-share`** across reboot. The NFS server `servera.lab.example.com` exports the **`/share`** network file system. The NFS export path is `servera.lab.example.com:/share`.
- Create a new 512 MiB partition on the `/dev/vdc` disk to be used as swap space. This swap space must be automatically activated at boot.
- Create a new group called `production`. Create the `production1`, `production2`, `production3`, and `production4` users. Ensure that they use the new group called `production` as their supplementary group.

- Configure your system so that it uses a new directory called **/run/volatile** to store temporary files. Files in this directory should be subject to time based cleanup if they are not accessed for more than 30 seconds. The octal permissions for the directory must be **0700**. Make sure that you use the **/etc/tmpfiles.d/volatile.conf** file to configure the time based cleanup for the files in **/run/volatile**.

- Create the new directory called **/webcontent**. Both the owner and group of the directory should be `root`. The group members of `production` should be able to read and write to this directory. The `production1` user should only be able to read this directory. These permissions should apply to all new files and directories created under the **/webcontent** directory.

1. From `workstation`, open an SSH session to `serverb` as `student`.

```
[student@workstation ~]$ ssh student@serverb
...output omitted...
```

2. Switch to the `root` user.

```
[student@serverb ~]$ sudo -i
[sudo] password for student: student
[root@serverb ~]#
```

3. Create a 2 GiB partition on `/dev/vdb`.

```
[root@serverb ~]# parted /dev/vdb mklabel msdos
[root@serverb ~]# parted /dev/vdb mkpart primary 1GiB 3GiB
```

4. Create a logical volume called `vol_home` using the 2 GiB partition you created on `/dev/vdb`. Name the volume group `extra_storage`.

   4.1. Declare the `/dev/vdb1` block device as a physical volume.

```
[root@serverb ~]# pvcreate /dev/vdb1
...output omitted...
```

   4.2. Create the `extra_storage` volume group using `/dev/vdb1`.

```
[root@serverb ~]# vgcreate extra_storage /dev/vdb1
...output omitted...
```

   4.3. Create a 1 GiB logical volume named `vol_home`.

```
[root@serverb ~]# lvcreate -L 1GiB -n vol_home extra_storage
...output omitted...
```

5. Format `vol_home` with the XFS file-system type, and mount it on **/home-directories**.

   5.1. Create a directory called **/home-directories**.

```
[root@serverb ~]# mkdir /home-directories
```

5.2. Format `/dev/extra_storage/vol_home` with the XFS file-system type.

```
[root@serverb ~]# mkfs -t xfs /dev/extra_storage/vol_home
...output omitted...
```

5.3. Persistently mount `/dev/extra_storage/vol_home` on **`/home-directories`**. Use the structure's UUID when creating the entry in **`/etc/fstab`**.

```
[root@serverb ~]# lsblk -o UUID /dev/extra_storage/vol_home
UUID
988cf149-0667-4733-abca-f80c6ec50ab6
[root@serverb ~]# echo "UUID=988c...0ab6 /home-directories \
xfs defaults 0 0" >> /etc/fstab
[root@serverb ~]# mount -a
```

6. Ensure that the network file system called **`/share`** is persistently mounted on **`/local-share`** across reboot. The NFS server `servera.lab.example.com` exports the **`/share`** network file system. The NFS export path is `servera.lab.example.com:/share`.

6.1. Create the **`/local-share`** directory.

```
[root@serverb ~]# mkdir /local-share
```

6.2. Append the appropriate entry to **`/etc/fstab`** so that the network file system available at `servera.lab.example.com:/share` is persistently mounted on **`/local-share`** across reboot.

```
[root@serverb ~]# echo "servera.lab.example.com:/share /local-share \
nfs rw,sync 0 0" >> /etc/fstab
```

6.3. Mount the network file system on **`/local-share`** based on the entry in **`/etc/fstab`**.

```
[root@serverb ~]# mount /local-share
```

7. Create a new 512 MiB partition on the `/dev/vdc` disk to be used as swap space. This swap space must be automatically activated at boot time.

7.1. Create a 512 MiB partition on `/dev/vdc`.

```
[root@serverb ~]# parted /dev/vdc mklabel msdos
[root@serverb ~]# parted /dev/vdc mkpart primary 1MiB 513MiB
```

7.2. Make the swap space on `/dev/vdc1`.

```
[root@serverb ~]# mkswap /dev/vdc1
...output omitted...
```

7.3. Activate the swap space so that it persists across reboot. Use the structure's UUID when creating the entry in **`/etc/fstab`**.

```
[root@serverb ~]# lsblk -o UUID /dev/vdc1
UUID
cc18ccb6-bd29-48a5-8554-546bf3471b69
[root@serverb ~]# echo "UUID=cc18...1b69 swap \
swap defaults 0 0" >> /etc/fstab
[root@serverb ~]# swapon -a
```

8. Create the `production1`, `production2`, `production3`, and `production4` users. Ensure that they use the new group called `production` as their supplementary group.

```
[root@serverb ~]# groupadd production
[root@serverb ~]# for i in 1 2 3 4; do useradd -G production production$i;
 done
```

9. Configure your system so that it uses a new directory called **`/run/volatile`** to store temporary files. Files in this directory should be subject to time based cleanup if they are not accessed for more than 30 seconds. The octal permissions for the directory must be **`0700`**. Make sure that you use the **`/etc/tmpfiles.d/volatile.conf`** file to configure the time based cleanup for the files in **`/run/volatile`**.

9.1. Create a file called **`/etc/tmpfiles.d/volatile.conf`** with the following content.

```
d /run/volatile 0700 root root 30s
```

9.2. Use the **`systemd-tmpfiles --create`** command to create the **`/run/volatile`** directory if it does not exist.

```
[root@servera ~]# systemd-tmpfiles --create /etc/tmpfiles.d/
volatile.conf
```

10. Create a new directory called **`/webcontent`**. Both the owner and group owner of the directory should be `root`. The group members of `production` should be able to read and write to this directory. The `production1` user should only be able to read this directory.

These permissions should apply to all new files and directories created under the **/webcontent** directory.

10.1. Create the **/webcontent** directory.

```
[root@serverb ~]# mkdir /webcontent
```

10.2. Use **setfacl** to configure permissions on **/webcontent** so that the group members of `production` have both read and write permissions to it, with the exception of the `production1` user, who should only be granted read permission.

```
[root@serverb ~]# setfacl -m u:production1:rx /webcontent
[root@serverb ~]# setfacl -m g:production:rwx /webcontent
```

10.3. Use **setfacl** to set the default permissions on **/webcontent** so that the permissions you applied in the preceding step also apply to all new files and directories created under the **/webcontent** directory.

```
[root@serverb ~]# setfacl -m d:u:production1:rx /webcontent
[root@serverb ~]# setfacl -m d:g:production:rwx /webcontent
```

10.4. Exit the `root` user's shell.

```
[root@serverb ~]# exit
logout
```

10.5. Log off from `serverb`.

```
[student@serverb ~]$ exit
logout
Connection to serverb closed.
```

Evaluation

On `workstation`, run the **lab rhcsa-compreview2 grade** script to confirm success on this exercise. Correct any reported failures and rerun the script until successful.

```
[student@workstation ~]$ lab rhcsa-compreview2 grade
```

Finish

On `workstation`, run **lab rhcsa-compreview2 finish** to complete this exercise. This script deletes the files and resources created throughout the exercise and ensures that the environment is clean.

```
[student@workstation ~]$ lab rhcsa-compreview2 finish
```

This concludes the comprehensive review.

LAB

CONFIGURING AND MANAGING SERVER SECURITY

In this review, you will configure SSH key-based authentication, change firewall settings, adjust the SELinux mode and an SELinux Boolean, and troubleshoot SELinux issues.

OUTCOMES

You should be able to:

- Configure SSH keys for key-based authentication.
- Configure firewall settings.
- Adjust the SELinux mode and SELinux Booleans.
- Troubleshoot SELinux issues.

BEFORE YOU BEGIN

Log in to `workstation` as `student` using `student` as the password.

On `workstation`, run **`lab rhcsa-compreview3 start`** to start the comprehensive review. This script creates the necessary files to set up the environment correctly.

```
[student@workstation ~]$ lab rhcsa-compreview3 start
```

INSTRUCTIONS

Perform the following tasks to complete the comprehensive review:

- Generate SSH keys for the `student` user on `serverb`. Do not protect the private key with a passphrase.
- On `servera`, configure the `student` user to accept login authentication using the SSH key pair created for `student` on `serverb`. The `student` user on `serverb` should be able to log in to `servera` using SSH without entering a password. Use `student` as the password of the `student` user, if required.
- On `servera`, change the default SELinux mode to **`permissive`**.
- Configure `serverb` to automatically mount the home directory of the `production5` user when the user logs in, using the network file system **`/home-directories/production5`**. This network file system is exported from `servera.lab.example.com`. Adjust the appropriate SELinux Boolean so that `production5` can use the NFS-mounted home directory on `serverb` after authenticating via SSH key-based authentication. The `production5` user's password is `redhat`.
- On `serverb`, adjust the firewall settings so that the SSH connections originating from `servera` are rejected.

- On `serverb`, investigate and fix the issue with the Apache HTTPD daemon, which is configured to listen on port `30080/TCP`, but which fails to start. Adjust the firewall settings appropriately so that the port `30080/TCP` is open for incoming connections.

Evaluation

On `workstation`, run the **`lab rhcsa-compreview3 grade`** script to confirm success on this exercise. Correct any reported failures and rerun the script until successful.

```
[student@workstation ~]$ lab rhcsa-compreview3 grade
```

Finish

On `workstation`, run **`lab rhcsa-compreview3 finish`** to complete this exercise. This script deletes the files and resources created throughout the exercise and ensures that the environment is clean.

```
[student@workstation ~]$ lab rhcsa-compreview3 finish
```

Save any files or work you want to keep to other systems, and then reset `workstation`, `servera`, and `serverb`.

This concludes the comprehensive review.

SOLUTION

CONFIGURING AND MANAGING SERVER SECURITY

In this review, you will configure SSH key-based authentication, change firewall settings, adjust the SELinux mode and an SELinux Boolean, and troubleshoot SELinux issues.

OUTCOMES

You should be able to:

- Configure SSH keys for key-based authentication.
- Configure firewall settings.
- Adjust the SELinux mode and SELinux Booleans.
- Troubleshoot SELinux issues.

BEFORE YOU BEGIN

Log in to `workstation` as `student` using `student` as the password.

On `workstation`, run **`lab rhcsa-compreview3 start`** to start the comprehensive review. This script creates the necessary files to set up the environment correctly.

```
[student@workstation ~]$ lab rhcsa-compreview3 start
```

INSTRUCTIONS

Perform the following tasks to complete the comprehensive review:

- Generate SSH keys for the `student` user on `serverb`. Do not protect the private key with a passphrase.
- On `servera`, configure the `student` user to accept login authentication using the SSH key pair created for `student` on `serverb`. The `student` user on `serverb` should be able to log in to `servera` using SSH without entering a password. Use `student` as the password of the `student` user, if required.
- On `servera`, change the default SELinux mode to **`permissive`**.
- Configure `serverb` to automatically mount the home directory of the `production5` user when the user logs in, using the network file system **`/home-directories/production5`**. This network file system is exported from `servera.lab.example.com`. Adjust the appropriate SELinux Boolean so that `production5` can use the NFS-mounted home directory on `serverb` after authenticating via SSH key-based authentication. The `production5` user's password is `redhat`.
- On `serverb`, adjust the firewall settings so that the SSH connections originating from `servera` are rejected.

- On `serverb`, investigate and fix the issue with the Apache HTTPD daemon, which is configured to listen on port `30080/TCP`, but which fails to start. Adjust the firewall settings appropriately so that the port `30080/TCP` is open for incoming connections.

1. From `workstation`, open an SSH session to `serverb` as `student`.

```
[student@workstation ~]$ ssh student@serverb
...output omitted...
```

2. Generate SSH keys for the `student` user on `serverb` using the **`ssh-keygen`** command. Do not protect the private key with a passphrase.

```
[student@serverb ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/student/.ssh/id_rsa): Enter
Created directory '/home/student/.ssh'.
Enter passphrase (empty for no passphrase): Enter
Enter same passphrase again: Enter
Your identification has been saved in /home/student/.ssh/id_rsa.
Your public key has been saved in /home/student/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:1TPZ4TXYwiGWfExUGtRTHgfKQbF9hVuLa+VmH4vgkFY
 student@serverb.lab.example.com
The key's randomart image is:
+---[RSA 2048]----+
|          .+@BO**|
|          .=.#+B*|
|          . X.*o=|
|         . E +.+ |
|        S o   +  |
|         + . o = |
|        . o o + +|
|           . . ..|
|                 |
+----[SHA256]-----+
```

3. On `servera`, configure the `student` user to accept login authentication using the SSH key pair you created for `student` on `serverb`. The `student` user on `serverb` should be able to log in to `servera` using SSH without entering a password. Use `student` as the password of the `student` user, when required.

   3.1. Use the **`ssh-copy-id`** command to transfer the public key of the SSH key pair of `student` on `serverb` to `student` on `servera`. Use `student` as the password of the `student` user, if prompted.

```
[student@serverb ~]$ ssh-copy-id student@servera
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/
student/.ssh/id_rsa.pub"
The authenticity of host 'servera (172.25.250.10)' can't be
 established.
```

```
ECDSA key fingerprint is SHA256:g/
fIMtVzDWTbTi1l0OwC30sL6cHmro9Tf563NxmeyyE.
Are you sure you want to continue connecting (yes/no)? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s),
 to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you
 are prompted now it is to install the new keys
student@servera's password: student

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'student@servera'"
and check to make sure that only the key(s) you wanted were added.
```

3.2. Use the **ssh** command to verify that the `student` user can log in to `servera` from `serverb` without entering a password.

```
[student@serverb ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

4. On `servera`, change the default SELinux mode to **`permissive`**.

4.1. Edit **`/etc/sysconfig/selinux`** to set the value of the parameter **`SELINUX`** to **`permissive`**. You can use the **`sudo vi /etc/sysconfig/selinux`** command to edit the configuration file as the superuser. Use the password `student`, if prompted.

```
...output omitted...
#SELINUX=enforcing
SELINUX=permissive
...output omitted...
```

4.2. Use the **`sudo systemctl reboot`** command to reboot the system as the superuser.

```
[student@servera ~]$ sudo systemctl reboot
Connection to servera closed by remote host.
Connection to servera closed.
[student@serverb ~]$
```

5. Configure `serverb` to automatically mount the home directory of the `production5` user when the user logs in, using the network file system **`/home-directories/production5`**. This network file system is exported from `servera.lab.example.com`. Adjust the appropriate SELinux Boolean so that `production5` can use the NFS-mounted home

directory on `serverb` after authenticating via SSH key-based authentication. The `production5` user's password is `redhat`.

5.1. On `serverb`, use the **`sudo -i`** command to switch to the `root` user account.

```
[student@serverb ~]$ sudo -i
[sudo] password for student: student
[root@serverb ~]#
```

5.2. Install the *autofs* package.

```
[root@serverb ~]# yum install autofs
...output omitted...
Is this ok [y/N]: y
...output omitted...
Installed:
  autofs-1:5.1.4-29.el8.x86_64

Complete!
```

5.3. Create the `autofs` master map file called **`/etc/auto.master.d/production5.autofs`** with the following content.

```
/- /etc/auto.production5
```

5.4. Create the **`/etc/auto.production5`** file with the following content.

```
/localhome/production5 -rw servera.lab.example.com:/home-directories/
production5
```

5.5. Restart the `autofs` service.

```
[root@serverb ~]# systemctl restart autofs
```

6. On `servera`, verify that the `production5` user is not able to log in to `serverb` using SSH public-key authentication. An SELinux Boolean causes this issue which you will fix in the following steps.

6.1. From `workstation`, open an SSH session to `servera` as `student`.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

6.2. Switch to the `production5` user using the password `redhat`.

```
[student@servera ~]$ su - production5
Password: redhat
[production5@servera ~]$
```

6.3. Use the **`ssh-keygen`** command to generate the SSH keys as `production5`.

```
[production5@servera ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/production5/.ssh/
id_rsa): Enter
Created directory '/home/production5/.ssh'.
Enter passphrase (empty for no passphrase): Enter
Enter same passphrase again: Enter
Your identification has been saved in /home/production5/.ssh/id_rsa.
Your public key has been saved in /home/production5/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:zmin1nmCt4H8LA+4FPimtdg81nl7ATbInUFW3HSPxk4
 production5@servera.lab.example.com
The key's randomart image is:
+---[RSA 2048]----+
|       .oo.o. .  |
|       .. . .o o |
|     . o o    E .|
|   .  o *    +   |
|  . .  .So    .  |
|   . + =  .      |
```

```
|     *.*+=. .      |
|    Oo+***.o       |
|   o.=o.=**        |
+----[SHA256]-----+
```

6.4. Use the **ssh-copy-id** command to transfer the public key of the SSH key pair of production5 on servera to production5 on serverb. Use redhat as the password of the production5 user, if prompted.

```
[production5@servera ~]$ ssh-copy-id production5@serverb
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/
production5/.ssh/id_rsa.pub"
The authenticity of host 'serverb (172.25.250.11)' can't be
 established.
ECDSA key fingerprint is
 SHA256:ciCkaRWF4g6eR9nSdPxQ7KL8czpViXal6BousK544TY.
Are you sure you want to continue connecting (yes/no)? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s),
 to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you
 are prompted now it is to install the new keys
production5@serverb's password: redhat

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'production5@serverb'"
and check to make sure that only the key(s) you wanted were added.
```

6.5. Use the SSH public key-based authentication instead of password-based authentication to log in to serverb as production5. This command should fail.

```
[production5@servera ~]$ ssh -o pubkeyauthentication=yes \
-o passwordauthentication=no production5@serverb
production5@serverb: Permission denied (publickey,gssapi-keyex,gssapi-
with-mic,password).
```

7. Set the appropriate SELinux Boolean setting on serverb, so that production5 can log in to serverb using the SSH public key-based authentication and use the home directory.

7.1. On serverb as root, set the use_nfs_home_dirs SELinux Boolean to **true**.

```
[root@serverb ~]# setsebool -P use_nfs_home_dirs true
```

7.2. Use the SSH public key-based authentication instead of password-based authentication to log in to serverb as production5. This command should succeed.

```
[production5@servera ~]$ ssh -o pubkeyauthentication=yes \
-o passwordauthentication=no production5@serverb
...output omitted...
[production5@serverb ~]$
```

8. On `serverb`, adjust the firewall settings so that SSH connections originating from `servera` are rejected. The `servera` system uses the IPv4 address `172.25.250.10`.

8.1. Use the **`firewall-cmd`** command to add the IPv4 address of `servera` to the `firewalld` zone called `block`.

```
[root@serverb ~]# firewall-cmd --add-source=172.25.250.10/32 \
--zone=block --permanent
success
```

8.2. Use the **`firewall-cmd --reload`** command to reload the changes in the firewall settings.

```
[root@serverb ~]# firewall-cmd --reload
success
```

9. On `serverb`, investigate and fix the issue with the Apache HTTPD daemon, which is configured to listen on port `30080/TCP`, but which fails to start. Adjust the firewall settings appropriately so that port `30080/TCP` is open for incoming connections.

9.1. Use the **`systemctl`** command to restart the `httpd` service. This command fails to restart the service.

```
[root@serverb ~]# systemctl restart httpd.service
Job for httpd.service failed because the control process exited with
 error code.
See "systemctl status httpd.service" and "journalctl -xe" for details.
```

9.2. Use the **`systemctl status`** command to investigate the reason for the failure of the `httpd` service.

```
[root@serverb ~]# systemctl status httpd.service
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled;
 vendor preset: disabled)
   Active: failed (Result: exit-code) since Mon 2019-04-15 06:42:41
 EDT; 5min ago
     Docs: man:httpd.service(8)
  Process: 27313 ExecStart=/usr/sbin/httpd $OPTIONS -DFOREGROUND
 (code=exited, status=1/FAILURE)
 Main PID: 27313 (code=exited, status=1/FAILURE)
   Status: "Reading configuration..."

Apr 15 06:42:41 serverb.lab.example.com systemd[1]: Starting The Apache
 HTTP Server...
Apr 15 06:42:41 serverb.lab.example.com httpd[27313]: (13)Permission
 denied: AH00072: make_sock: could not bind to address [::]:30080
Apr 15 06:42:41 serverb.lab.example.com httpd[27313]: (13)Permission
 denied: AH00072: make_sock: could not bind to address 0.0.0.0:30080
Apr 15 06:42:41 serverb.lab.example.com httpd[27313]: no listening
 sockets available, shutting down
```

```
Apr 15 06:42:41 serverb.lab.example.com httpd[27313]: AH00015: Unable
 to open logs
Apr 15 06:42:41 serverb.lab.example.com systemd[1]: httpd.service: Main
 process exited, code=exited, status=1/FAILURE
Apr 15 06:42:41 serverb.lab.example.com systemd[1]: httpd.service:
 Failed with result 'exit-code'.
Apr 15 06:42:41 serverb.lab.example.com systemd[1]: Failed to start The
 Apache HTTP Server.
```

Notice the permission error in the preceding output, which signifies that the `httpd` daemon failed to bind to port `30080/TCP`. The SELinux policy can be a potential restriction for an application to bind to a port. Press **q** to quit the preceding **`systemctl`** command.

9.3. Use the **`sealert`** command to determine if an SELinux policy is preventing `httpd` from binding to port `30080/TCP`.

```
[root@serverb ~]# sealert -a /var/log/audit/audit.log
100% done
found 1 alerts in /var/log/audit/audit.log
--------------------------------------------------------------------------------

SELinux is preventing /usr/sbin/httpd from name_bind access on the
 tcp_socket port 30080.

*****  Plugin bind_ports (92.2 confidence) suggests
 ************************

If you want to allow /usr/sbin/httpd to bind to network port 30080
Then you need to modify the port type.
Do
# semanage port -a -t PORT_TYPE -p tcp 30080
    where PORT_TYPE is one of the following: http_cache_port_t,
 http_port_t, jboss_management_port_t, jboss_messaging_port_t,
 ntop_port_t, puppet_port_t.
```

The preceding log message reveals that the port `30080/TCP` does not have the appropriate SELinux context **`http_port_t`**, causing SELinux to prevent `httpd` to

bind to this port. The log message also produces the syntax of the **semanage port** command so that you can easily fix the issue.

9.4. Use the **semanage port** command to set the appropriate SELinux context on the port 30080/TCP for httpd to bind to it.

```
[root@serverb ~]# semanage port -a -t http_port_t -p tcp 30080
```

9.5. Use the **systemctl** command to restart httpd. This command should successfully restart the service.

```
[root@serverb ~]# systemctl restart httpd
```

9.6. Add the port 30080/TCP to the default firewalld zone called public.

```
[root@serverb ~]# firewall-cmd --add-port=30080/tcp --permanent
success
[root@serverb ~]# firewall-cmd --reload
success
```

9.7. Exit the root user's shell.

```
[root@serverb ~]# exit
logout
```

9.8. Log off from serverb.

```
[student@serverb ~]$ exit
logout
Connection to serverb closed.
```

Evaluation

On workstation, run the **lab rhcsa-compreview3 grade** script to confirm success on this exercise. Correct any reported failures and rerun the script until successful.

```
[student@workstation ~]$ lab rhcsa-compreview3 grade
```

Finish

On workstation, run **lab rhcsa-compreview3 finish** to complete this exercise. This script deletes the files and resources created throughout the exercise and ensures that the environment is clean.

```
[student@workstation ~]$ lab rhcsa-compreview3 finish
```

Save any files or work you want to keep to other systems, and then reset workstation, servera, and serverb.

This concludes the comprehensive review.